OUTILS LOGICIELS POUR LA SURVEILLANCE ET LA DÉTECTION DES FAUTES DANS LES SYSTÈMES DISTRIBUÉS

par

Hatem Jedidi

mémoire présenté au Département de mathématiques et d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, juin 1997

Sommaire

La conduite d'un procédé distribué dynamique nécessite la prise en compte et le traitement élaboré d'informations de natures diverses. Pour ce faire, nous avons développé un système de simulation et de contrôle, appelé MSS, capable de conseiller l'opérateur et de l'alerter en cas d'apparition de fautes ou de dysfonctionnement.

MSS se présente comme un outil graphique pouvant opérer sur des entités qui sont en partie l'image des objets manipulés par le procédé réel, avec quelques extensions. Ces entités sont appelées **objets MSS** et leurs représentations graphiques ainsi que les relations possibles entre elles sont appelées **primitives MSS**. Il s'agit d'un système temps-réel que l'on peut décrire comme un ensemble de tâches coopérantes s'exécutant en parallèle et soumises à des contraintes temporelles souvent strictes. Pour leur mise en oeuvre effective, nous nous sommes basés sur la décomposition de l'application en tâches (entités actives) coopérantes traitant les fonctionnalités de l'application. Celles-ci sont implantées sous forme de tâches dont l'exécution peut être différée. Leur gestion est confiée à un noyau de synchronisation qui a pour objet de prendre en compte l'automatisation des événements tels que les suspensions de processus et les arrêts d'urgence, de même que certaines activités réflexes de l'automatisme à concevoir. Ces traitements constituent la partie réactive de l'automatisme.

Remerciements

Je voudrais exprimer tout d'abord mes remerciements aux professeurs qui m'ont fait l'honneur de participer au jury de ce mémoire.

Je remercie spécialement le professeur Béchir El Ayeb, mon directeur de recherche, qui m'a appuyé par ses conseils, son soutien, sa grande disponibilité et la confiance qu'il m'a accordée durant toute la période de la recherche. Ceci a été d'une aide précieuse afin d'orienter et de bien guider mon travail. Qu'il trouve ici l'expression de ma gratitude pour l'intérêt qu'il porte à ce travail.

Je tiens aussi à exprimer mes remerciements à mon collègue Monsieur Lotfi Ben Romdhane pour les nombreuses discussions que nous avons eues ensemble et qui ont guidé mes travaux, ainsi que pour l'environnement de travail que nous avons créé au sein du laboratoire.

Je remercie enfin ma famille et tous les amis qui m'ont soutenu et aidé pour l'aboutissement de mon travail.

TABLE DES MATIÈRES

LISTE DES FIGURES

Introduction

L'ère courante peut être décrite comme étant l'ère de l'automatisation. Ceci est dû à l'intérêt croissant porté à l'analyse et au contrôle des systèmes durant les dernières décennies.

Une large gamme de processus rencontrés par des ingénieurs de différents domaines (électronique, mécanique, industriel), par des chercheurs et des scientifiques et par des économistes peut être vue selon la formulation suivante :
Considérons un système physique où le terme "physique" est utilisé dans un sens suffisamment large pour couvrir tout processus provenant du monde réel. Ce système peut être décrit à n'importe quel instant t, par un simple ensemble de variables quantifiables $x_1(t), x_2(t), ..., x_n(t)$. Ces variables, nommées **variables d'état**, constituent les composantes d'un vecteur $x(t)$, le vecteur d'état.

Dans plusieurs cas, il suffit que $x(t)$ soit unidimensionnel. En effet, celui-ci peut représenter la position d'une particule sur une ligne, l'altitude d'un satellite ou une température dans un four. Mais généralement, $x(t)$ est considéré comme un vecteur multidimensionnel. Par exemple, dans les circuits électriques, les composantes de x peuvent représenter le courant et le voltage; dans un processus chimique, elles peuvent représenter des concentrations et des températures, et enfin, dans un système économique, elles peuvent représenter les capacités productives et le stock.

Comme nous l'avons indiqué par la notation et les exemples, nous sommes intéressés par les systèmes dynamiques plutôt que les systèmes statiques.

Dans le domaine industriel, les systèmes de conduite et de surveillance analogique sont relativement coûteux et se prêtent mal à la mise en oeuvre d'algorithmes complexes susceptibles d'être modifiés au cours du temps. De plus, ces systèmes ne peuvent que difficilement être multiplexés et ils conduisent à des solutions coûteuses pour l'implantation des opérations liées à la conduite, comme par exemple l'enregistrement de l'historique de l'évolution du processus. De ce fait, la tendance actuelle consiste à effectuer toutes les opérations de pilotage et de surveillance par ordinateur. Cette évolution vers la numérisation doit prendre en compte aussi bien les avantages que les inconvénients des approches classiques de contrôle. Une solution, pour une meilleure intégration de l'ordinateur dans la surveillance des procédés dynamiques, consiste à développer un outil graphique qui facilite l'intervention de l'opérateur en remédiant aux inconvénients rencontrés par l'utilisation des tableaux de bord analogiques, tout en conservant les avantages de ceux-ci. L'exploitation d'un système distribué est grandement simplifiée lorsque les fonctions de supervision et de commande sont rassemblées de façon à pouvoir piloter tout le système à partir d'un seul poste de pilotage. Cette disposition permet de réduire le nombre de personnes affectées à l'exploitation et de réagir plus rapidement en cas d'évolution anormale du processus.

Compte tenu de toutes ces motivations, faisant abstraction des notions de décentralisation et d'interactions entre le système et l'opérateur comme entre les composantes du système lui-même, nous avons exprimé le souhait de pouvoir disposer d'un outil graphique évolutif d'expertise et d'aide à la conduite. Cet outil devrait permettre d'élaborer des réponses aux questions suivantes : étant donné un système distribué quelconque, quelle organisation spatiale définir, quelle technique utiliser et quels objectifs viser pour exploi-

ter au mieux les possibilités de ce dernier tout en profitant au maximum des capacités de calcul des ordinateurs, ainsi que de la performance des techniques existantes?

Au terme d'une première étape de réflexion et d'analyse sur les systèmes, les techniques et les besoins dans le domaine de la surveillance, nous avons proposé le développement d'un système de simulation et de contrôle nommé MSS (**M**onitoring and **S**imulation **S**ystem) par la mise en oeuvre de la démarche générale suivante, structurée en trois chapitres principaux.

Le premier chapitre intitulé "le prototype MSS", consiste à présenter la structure interne du système et à décrire sa structure hiérarchique, qui permet de lui donner un aspect proche de la réalité tout en détaillant les différents moyens d'interaction mentionnés ci-dessus. Le chapitre suivant nommé "l'interface graphique", vise à présenter MSS comme un outil qui offre plusieurs facilités d'exploitation du système, qui englobe aussi bien les ressources matérielles que logicielles. Le dernier chapitre "expérimentation et extension" s'intéresse à l'expérimentation de l'outil en réalisant quelques applications dans différents domaines industriels.

CHAPITRE 1

Le Prototype MSS

Objectif

La conception et la réalisation d'une application temps réel industrielle présentent plusieurs difficultés. Le prototype MSS constitue un outil graphique de simulation et de contrôle associé à un processus physique évoluant dans le temps.[1]
Au niveau conception, le découpage de MSS en deux sous-systèmes est indispensable, distinguant ainsi le sous-système *Transformationnel* (supervision) d'un autre, *Réactif* (dispositifs) [7].

Dans ce chapitre nous consacrerons exclusivement notre attention sur les détails des deux sous-systèmes de MSS. Nous allons aborder ensuite les outils mis en oeuvre afin d'assurer un ordonnancement cohérent des tâches qu'une application est supposée accomplir. Enfin nous allons mettre à la disposition de l'opérateur les éléments indispensables pour gérer

1. Le lecteur intéressé par les fondements théoriques et méthodologiques sous-jacents à MSS peut se référer à [3]

les fonctions nécessaires à la manipulation des objets dans le sens où un objet représente toute entité caractérisée par un état et un ensemble de fonctions qui opèrent sur celui ci. Le terme objet peut représenter un objets MSS, une application ou un environnement.

Dans ce qui suit nous allons utiliser les définitions suivantes.

Objet : c'est la composante élémentaire dans MSS, capable de communiquer avec le monde extérieur.

Application : c'est le processus physique à simuler, il est défini par un schéma constitué des objets placés dans un ordre défini et possédant un comportement spécifique permettant de simuler le processus réel. Ce comportement est le résultat des communications entre les objets existants dans cette application.

Environnement : c'est le domaine que l'utilisateur de MSS choisit pour définir, simuler et contrôler son *application.*

Un environnement est défini par une bibliothèque d'objets ayant une caractéristique principale commune qui est le **domaine**, un ensemble d'applications construites à partir des objets disponibles, et enfin une structure graphique, présentant les facilités d'exploitation et de manipulation des objets ainsi que des applications.

1.1 Le système réactif

Dans la conception de l'outil MSS, nous avons adopté l'approche objet. Nous avons conçus une structure hiérarchique ayant plusieurs niveaux d'abstraction. Ainsi le système réactif a été décomposé en trois sous-systèmes nommés **niveaux**, présentant des liens de type **père-fils**, et capables de produire des réactions de type **réflexe** [2] (voir figure 1.1).

1. Niveau objet
2. Niveau application

3. Niveau environnement

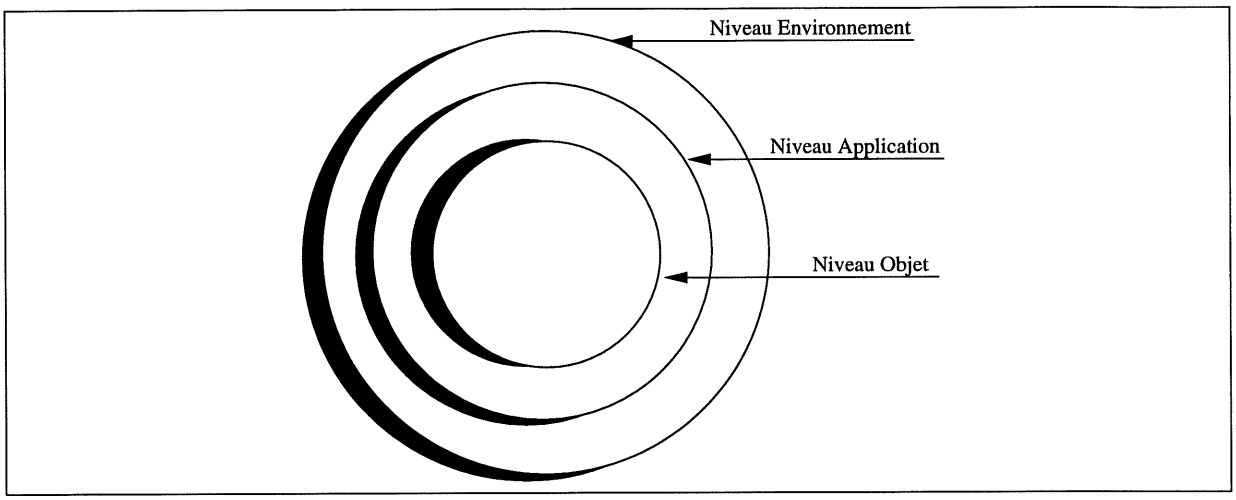

Figure 1.1 – *Les trois niveaux de MSS*

1.1.1 Le niveau objet

Un objet représente l'élément de base du *niveau Objet.* C'est une entité identifiable possédant un certain nombre de variables décrivant son état et des méthodes ou fonctions déterminant son comportement. L'interaction entre deux objets se limite à l'échange de messages ou valeurs, à la suite de quoi un objet peut déterminer son état ultérieur ou son comportement futur.

La conception des objets

L'interrelation entre objets, illustrée par la figure 1.2, montre une souplesse et une facilité de transformation et d'interaction, suffisantes pour que ces derniers puissent âccomplir leurs tâches au sein du système.
La structure interne d'un objet est présentée dans la figure 1.3. Nous pouvons remarquer

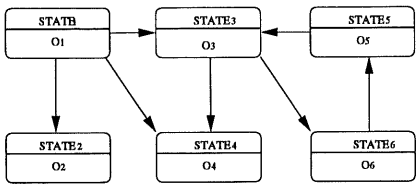

Figure 1.2 – *Interactions entre objets*

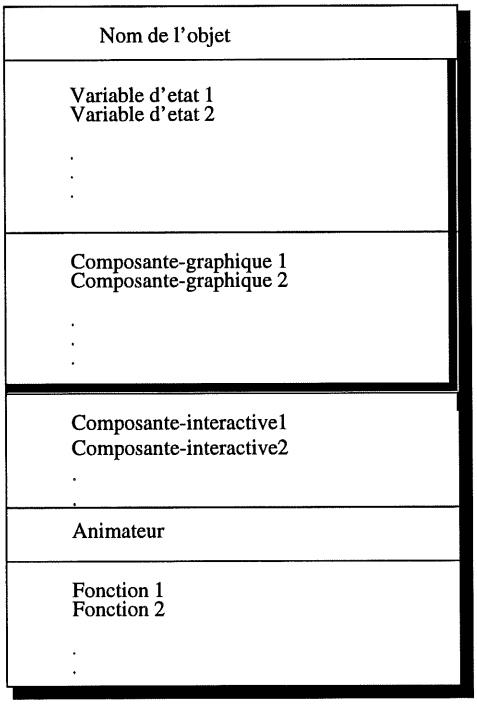

Figure 1.3 – *Structure d'un Objet*

qu'un objet possède une structure classique, qui est la suivante :
OBJ=<NOM, ÉTAT, COMPORTEMENT>.

Le nom: Les objets sont organisés sous une structure hiérarchique. Le niveau supérieur, est composé par les objets "père", qui possèdent des noms, représentant la famille ou le nom réel du dispositif physique. Les objets descendants de cette famille possèdent un nom interne, généré automatiquement par l'outil de développement. Ces noms sont en général composés par le nom du père suivi par une succession de chiffres.

Les variables d'état: comme leur nom l'indique, elles décrivent l'état d'un objet. L'état est simplement un ensemble de paramètres décrivant des entités réelles; e.g. entrée, sortie, température, pression.

Suivant le rôle qu'elles assument, les variables d'état peuvent être classées en trois catégories:

1. Les variables de configuration: elles dépendent essentiellement de l'application envisagée. Elles permettent aux objets d'interagir entre eux. A titre d'exemple, une variable d'état peut décrire les objets en entrée et ceux en sortie pour un objet donné.
2. Les variables de contrôle: chacune de celles-ci décrit une caractéristique physique de l'objet; e.g. température, pression, etc. C'est cet ensemble de variables que l'objet est supposé échanger comme messages avec les autres afin de refléter l'intérêt global du système simulé. À chaque variable de contrôle est affectée une constante ou un intervalle qui représente sa valeur idéale ou moyenne que cette dernière ne doit pas dépasser pour conserver un état non fautif. La notion de faute sera détaillée ultérieurement.

3. Les variables graphiques: c'est à ce niveau que l'apparence de l'objet intervient. Ces variables sont en réalité de nature complexe d'où leurs nomination "composantes graphiques". En général un objet est construit à partir d'une icône associée à d'autres objets graphiques élémentaires (cercle, rectangle, ...). Les composantes doivent répondre à plusieurs critères de choix de façon à pouvoir former un objet qui se rapproche de la réalité et facile à animer.

Le comportement : c'est le mécanisme qui illustre les différentes âctivités de l'objet au cours de son évolution dans l'application. Il est présenté sous deux formes caractérisant les deux possibilités de communication: *Objet-Opérateur* et *Objet-Application.* Ces activités sont définies par les trois composantes suivantes:

1. Les composantes interactives: C'est l'ensemble des réactions de l'objet à des âctions provenant de l'opérateur. Nous pouvons donc définir cet ensemble comme étant une collection de fonctions permettant la communication *MSS-Opérateur.* Ces fonctions sont propres à l'objet, elle permettent en général de réagir à des actions de la souris ou du clavier.
2. L'animateur : c'est un module cyclique (**Loop**) de période réglable. Celle-ci est fixée suivant des contraintes surtout matérielles : vitesse de la machine, mémoire, etc ... L'animateur agit directement sur les variables d'état comme sur les composantes graphiques afin d'assurer la communication entre objets âinsi que l'évolution de ceux-ci dans le temps.
3. Les fonctions : rendu à cette étape, un objet est capable d'interagir avec d'autres objets ainsi qu'avec l'opérateur.

   Mise à part sa structure interne, un objet est le résultat d'une structure arborescente prenant racine âu niveau environnement, pour passer par le niveau application. Ce lien *Application-Objet* est caractérisé par un ensemble de fonctions capables d'agir sur les variables de configuration permettant ainsi de construire une application.

Ainsi, nous avons conçu des objets capables d'accomplir leurs tâches principales (communication et interaction) au sein de MSS. Cependant il faut mentionner que celui-ci ne possède aucun pouvoir de planification. Par contre nous pouvons lui assigner le terme "intelligent" vu son comportement réaliste fondé sur un échange de messages ou de valeurs lui permettant de fonctionner suivant le principe **stimuli-réponses** [4].

Les objets sont regroupés en environnements selon des caractéristiques communes; e.g. les variables de contrôle. Cependant, il existe des objets dont la présence est nécessaire dans tout environnement et qui représentent des services de lien du type *Objet-Objet* ou *Application-Objet.*

Les Objets de base

La nécessité de concevoir des liens physiques entre les composantes de MSS (Application et Objet) nous a conduit à créer un ensemble d'objets de base possédant une structure indépendante de tout environnement. Un objet de base est une entité ayant la structure d'objets décrite précédemment âinsi qu'une tâche ou un comportement bien défini.

MSS possède cinq objets de base que nous retrouvons dans chaque environnement (voir figure 1.4):

1. Le stream: c'est un moyen de créer un lien physique entre deux objets. En réalité un stream représente le canal d'acheminement des fluides ou des gaz dans les systèmes continus; e.g. l'industrie chimique et autres.

2. Le câble: il représente le moyen de lier tout objet de base à un autre. Le lien est physique, c'est l'équivalent d'un câble électrique puisque les objets de base représentent surtout des dispositifs électriques de contrôle.

3. Le capteur: c'est un dispositif défini par un module de détection ou de surveillance d'une variable de contrôle âssociée à un objet quelconque.

4. Le contrôleur: c'est l'équivalent d'un prédicat dans un programme. La tâche d'un contrôleur se limite à la surveillance d'une variable de contrôle associée à un objet, pour modifier certaines variables d'un autre objet, selon des contraintes définies dans son comportement.

5. Le simulateur de fautes: c'est simplement un objet capable de produire des valeurs ayant des influences sur une variable de contrôle dans l'objet considéré. Ces influences peuvent provoquer l'apparition d'une faute dans le fonctionnement global d'une application, (c'est un exemple de lien *Application-Objet*). Les valeurs produites sont habituellement communiquées âux objets par l'intermédiaire d'un tampon nommé **Channel**, détaillé dans la section suivante.

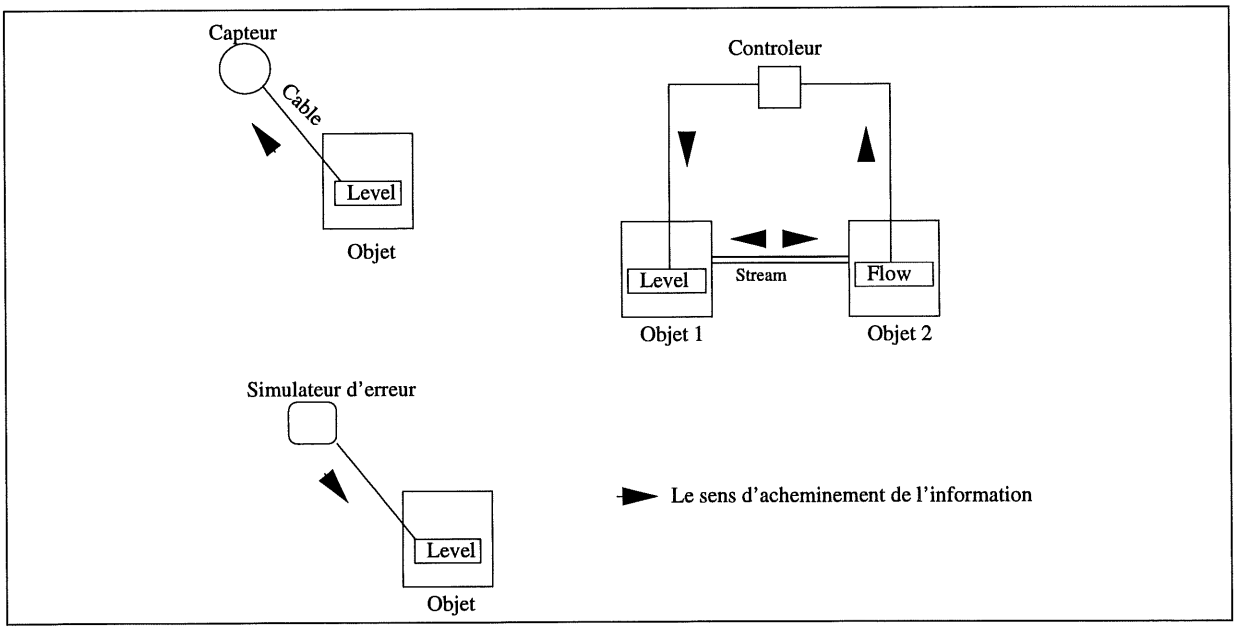

Figure 1.4 – *Les objets de base*

La communication entre objets

Le modèle multi-agents s'inspire du fonctionnement des systèmes de type stimuli-réponses. Un tel système est un ensemble organisé d'objets capables de réagir à des phénomènes externes déterminés (des stimuli) et de produire à leur tour des stimuli. Les paragraphes suivants exposent respectivement les principes directeurs du modèle de communication dans MSS ainsi que des moyens disponibles à sa mise en oeuvre.

L'aboutissement d'une coopération effective dans un système tel que MSS représente un problème difficile pour plusieurs raisons. La première est que le contrôle de décision d'un objet est basé seulement sur ses vues locales du problème global, ceci va mener à des décisions inappropriées sur le choix de la prochaine tâche à exécuter et sur l'identité de résultats doivent être transmis ou demandés à d'autres objets.

Nous avons choisi les "**streams**" comme étant le moyen de communication le plus adéquat entre les objets. En réalité un "stream" représente le plus souvent un canal de circulation des fluides. Mais, pour une conception modulaire générale, nous avons mis en oeuvre l'objet "**Channel**" comme deuxième moyen de communication.

Communication par échange de résultat intermédiaire

Afin d'éviter le problème d'inconsistance dans un système à base de connaissances, il est préférable d'échanger les résultats de haut niveau issus de l'interprétation de données de bas niveau. Cette résolution des ambiguïtés forme une partie intégrale de la tâche de résolution de problèmes de communication. Toutefois, permettre un libre échange d'information pour résoudre les ambiguïtés conduit rapidement à des coûts de communication excessifs [6].

Cette approche est de nature âscendante, ce qui implique une plus grande autonomie de prises de décision des objets. Les solutions sont produites au niveau de l'objet sans être influencées par les autres. Par contre, aucune âllocation de tâches ni planification n'est effectuée.

Dans les deux cas de communication **Objet-Objet** et **Objet-channel-Objet**, les objets se partagent des informations par le biais d'une base de connaissances. Celle-ci, dans le premier cas est représentée par le "**stream**". Il ne s'agit plus d'une seule base de connaissances mais, chaque deux objets en contact direct se partagent âu moins un "**stream**", où ils peuvent communiquer leurs variables d'état que l'on nomme "**messages**". Dans le second cas, une seule base de connaissances, nommée **Channel**, est partagée.

Objet-Objet: en fait si un objet est en communication avec un autre, âlors l'un des deux est nécessairement de type "stream" vu que ce dernier représente un lien physique existant. Un "stream" possède la structure d'un objet MSS capable de communiquer et de changer d'apparence, comme tout autre objet.

Chaque objet "reconnaît" un ensemble de "stream" en entrée et un autre en sortie. Celui-ci reçoit les informations de ses "streams" d'entrée, détermine ses nouvelles caractéristiques (variables d'état) et enfin communique ces dernières à l'ensemble des "streams" de sortie. (voir figure 1.5)

Les valeurs X1, X2... sont des variables temporaires utilisées pour permettre aux objets de sortie de déterminer leurs nouvelles variables d'état.

Objet-Channel-Objet: dans le même contexte d'échanges de variables d'état, ce moyen de communication a été élaboré pour produire une base de connaissances accessible à tous les objets présents dans l'application en question.

Un "Channel" est défini par un tampon comportant un ensemble d'entrées, chacune identifiée par une clé, représentant le nom de l'objet en question.
Une liste de noms des objets d'entrée et une autre pour ceux de sortie est mise à jour dans chaque objet impliqué, dans le but de leur permettre d'accéder aux informations stockées dans le "Channel". Ces informations sont aussi mises à jour à chaque cycle correspondant aux animateurs des objets.
Une fois sélectionné, l'objet n'a qu'à identifier l'information dont il a besoin en fournissant une clé représentée par le nom de la variable en question (voir figure 1.6).

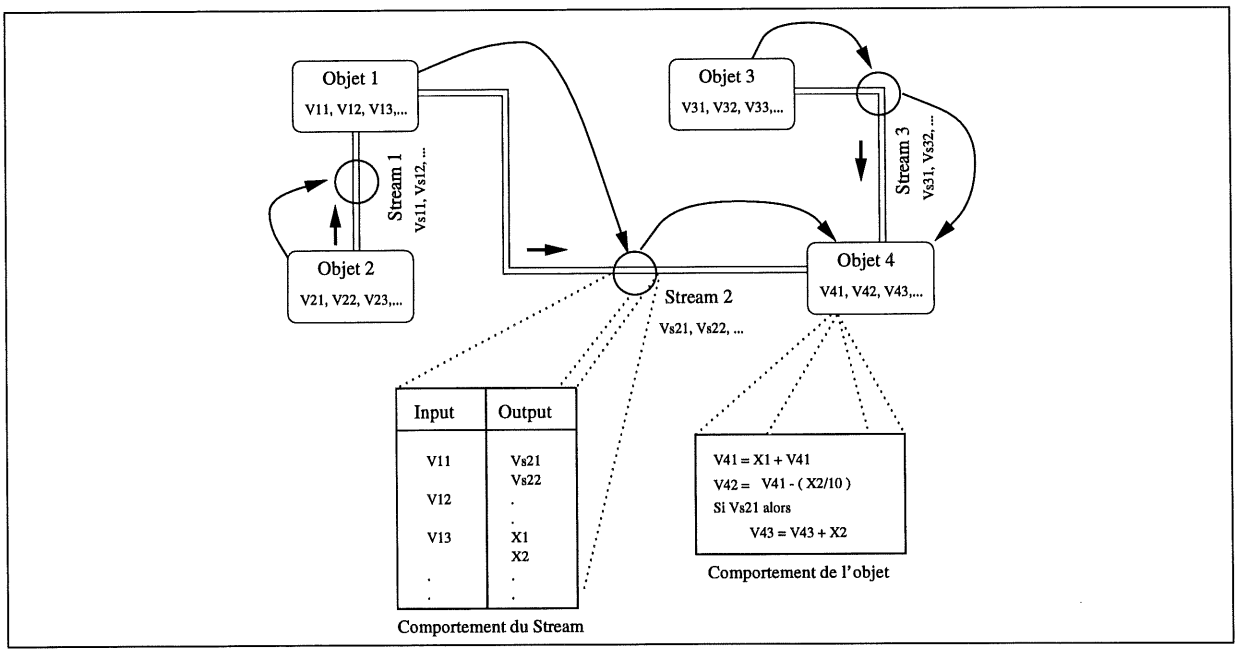

Figure 1.5 – *Communication Objet-Objet*

L'accès aux informations se fait par les fonctions **Put-Val** et **Get-Val** qui prennent en paramètres les deux clés mentionnées ci-dessus.

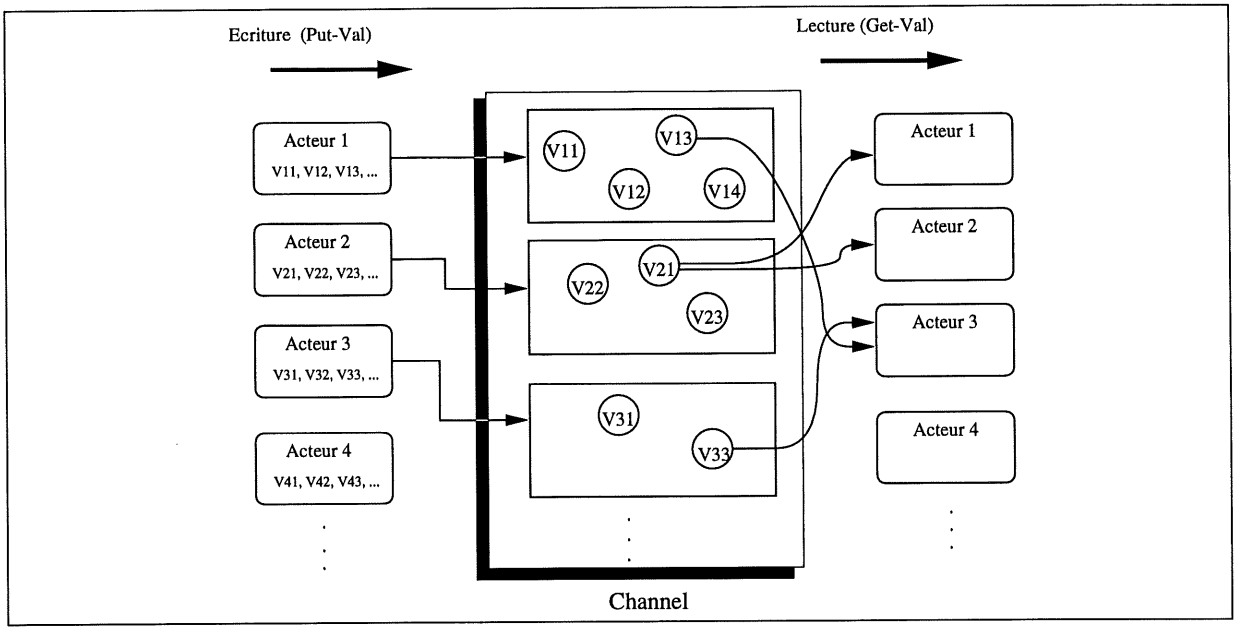

Figure 1.6 – *Communication Objet-Channel-Objet*

Exemple de communication:

Nous illustrons (figure 1.7) un exemple de communication entre deux objets crées par MSS. Cet exemple représente un cas très simple où les deux objets "Bassin" et "Tank"

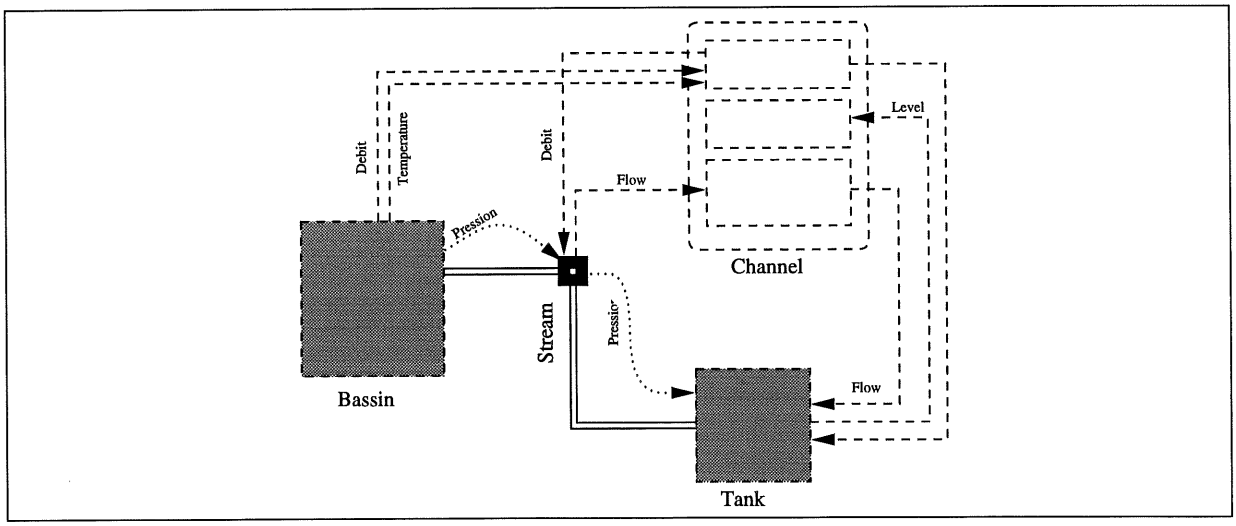

Figure 1.7 – *Un exemple de communication entre deux objets*

communiquent dans le but d'acheminer un liquide à travers un canal "stream".

L'application est constituée de quatre entités "Bassin", "Tank", "Stream", "Flow-controller". Les objets "Stream", et "Flow-controller" forment en réalité un objet unique, alors que les deux autres représentent les objets principaux de l'application. Les objets dessinés en pointillé sont invisibles à l'opérateur, leur présence n'est qu'illustrative.

Animée, cette application commence à évoluer dans le temps, chacun des trois objets impliqués dans l'application effectue les actions suivantes;

- accède au "Channel" par une clé qui est son nom afin de placer ses variables d'état. certaines variables sont placées dans le "stream" qui est le deuxième moyen de communication.
- accède aux champs dont il a besoin en utilisant les noms des objets représentant son "Input" ou son "Output". Un accès aux valeurs stockées dans le "stream" est aussi effectué en parallèle, représenté sur la figure pour le cas de la variable "pression".
- effectue un traitement ou un raisonnement qui lui permet de définir son apparence aux cycles suivants de simulation.

Ayant une structure complète, les objets, peuvent être groupés âfin de former une application, qui sera elle même considérée comme un objet.

1.1.2 Le niveau application

Comme au niveau objet, notre but est d'obtenir un ensemble d'outils permettant de manipuler des applications. Dans ce contexte nous avons conçu une structure pour les entités *application*, (voir figure 1.8), possédant les propriétés nécessaires pour répondre aux concepts de base de MSS.

Du point de vue opérateur, une application est un ensemble d'objets placés dans un ordre déterminé et capables de communiquer pour former un tout. En MSS une application est définie comme une séquence d'appels de fonctions, groupées en trois types.

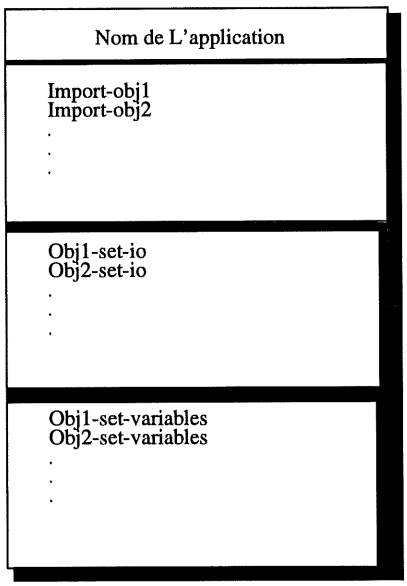

Figure 1.8 – *Structure d'une application*

Les fonctions de construction: c'est une série d'appels de fonctions du type "import-*obj*" définies dans le corps de l'objet considéré. Elles permettent de visualiser un objet à l'écran, i.e le placer à une position (x, y) prise comme paramètre. Ces fonctions permettent aussi de définir la liste d'objets qui forment initialement l'application. Les objets à ce stade, se retrouvent dans un état primitif, incapable de communiquer avec leur entourage.

Les fonctions de liaison: la communication entre les objets se fait par les canaux d'entrée/sortie. Il est donc nécessaire d'initialiser les variables de configuration relatives aux objets placés dans l'application.
Les fonctions de liaison consistent a une séquence d'appels de fonctions du type "*Obj*-Set-io".

Les fonctions de configuration: le dernier type de fonctions associé à une application concerne son initialisation. Ceci revient à fixer les caractéristiques de chaque objet impliqué dans l'application. L'appel de ce type de fonction se fait sous la forme "*Obj*-Set-Variables", ce qui permet d'affecter à chaque variable d'état de chaque objet formant l'application une valeur initiale, déterminant ainsi une configuration de l'application.

Ainsi, l'ensemble d'objets et d'applications engendre une structure que l'on appelle Environnement.

1.1.3 Le niveau environnement

Un environnement représente tout le système MSS spécialisé dans un domaine bien défini. Construire un environnement revient à préparer l'ensemble d'objets qui le caractérise âinsi que certaines applications si nécessaire. Un environnement, comme un objet, est une entité possédant un nom et un corps. La structure de son corps se rapproche de celle d'une application car elle est représentée sous forme d'une séquence d'appels de fonctions groupées en deux catégories (voir figure 1.9).

Les fonctions d'initialisation: C'est un ensemble d'appels de fonctions nécessaires pour préparer les variables ou caractéristiques de l'environnement âinsi que pour charger la bibliothèque d'objets qui lui est propre.

Les fonctions de configuration: Ici ce n'est plus un appel de fonction, mais des définitions de fonctions propres à l'environnement en cours. C'est simplement l'apparence de l'environnement.

En résumé, nous pouvons dire que le système **réactif** possède une structure à la fois souple, interactive et extensible. Ainsi, nous sommes prêts à décrire les mécanismes régissant l'interaction de ces composants, c'est à dire le système **transformationnel**.

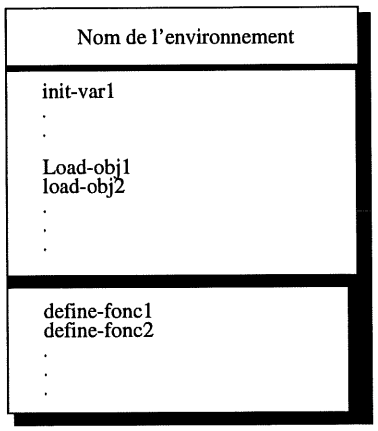

Figure 1.9 – *Structure d'un environnement*

1.2 Le système transformationnel

L'automatisation croissante des procédés industriels rend de plus en plus importante la surveillance en ligne de leur comportement [13].

MSS constitue un système de simulation et de contrôle de procédés physiques continus, ce qui couvre aussi bien la gestion de son fonctionnement idéal que le fonctionnement fautif. Pour cela il est nécessaire de prévoir un système capable d'organiser une scène

plus ou moins réaliste, d'où l'idée de l'utilisation d'une technique classique qui consiste à temporiser certaines actions. Si la durée prévue est terminée, une erreur est déclenchée.

1.2.1 Principe de simulation de fautes

Dans le contexte de construction d'un sous système de contrôle dans MSS, nous avons conçu un objet spécifique ayant le rôle de simulateur des fautes.
Le principe de fonctionnement de ce type d'objets est le suivant :

1. Une fois l'application construite et initialisée, les *simulateurs de fautes* sont placés en contact direct à travers l'objet "Câble" avec les objets jugés importants ou critiques. Chaque simulateur est associé à une variable d'état de l'objet correspondant.

2. Afin de lancer le processus de surveillance, l'application doit être d'abord animée. Une des tâches de ce processus consiste simplement à activer les *simulateurs de fautes*. Leur fonctionnement se réduit à modifier la variable d'état correspondante, c'est-à-dire la variable surveillée.

La surveillance dans MSS regroupe plusieurs fonctions en interaction (figure 1.10) que nous détaillons comme suit.

1. **Acquisition des données:** le système d'acquisition recueille des valeurs de capteurs. Ces derniers représentent un lien entre le système et le monde réel. Un capteur est un objet MSS comme les autres, possédant un comportement spécifique. Il se distingue principalement par:
   - Les variables **Object-to-control**: et **Slot-to-control** la première sert à reconnaître l'objet à surveiller, la deuxième identifie la variable d'état de l'objet surveillé. Celle-ci représente une donnée ayant un effet important sur le comportement de l'application en général. Pour cela il est nécessaire de savoir bien

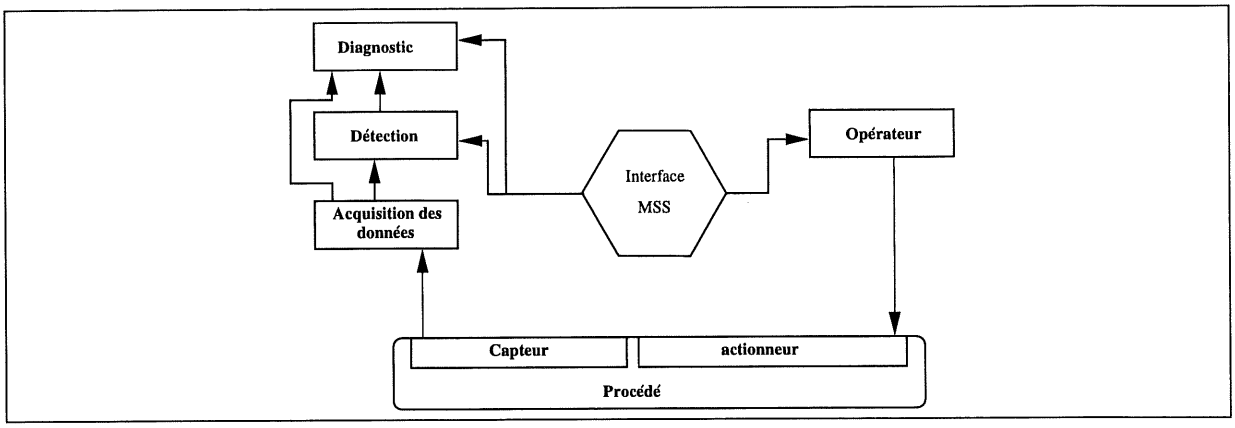

Figure 1.10 – *La structure du mécanisme de surveillance dans MSS*

placer ces capteurs, d'une façon minimale et efficace afin de réduire le coût de réalisation de l'application.

- Un comportement ou un animateur spécifique à cet objet et qui ne fait qu'afficher l'état de la variable à contrôler.

L'acquisition des données se fait à l'intérieur de l'objet lui même vu qu'il représente une **image** du procédé réel.

2. **Détection de fautes:** le module de détection englobe les moyens de détection des fautes et erreurs provenant du procédé en cours, il s'agit d'une détection directe. Dans son comportement, un objet possède un module d'auto-détection de fautes, basé sur la vérification de quelques variables d'état dites critiques. En parallèle, un mécanisme de détection de fautes est en fonctionnement périodique.
   L'information est ensuite passée au module *Diagnostic.*

3. **Diagnostic:** le module diagnostic sert à identifier et à localiser des fautes. Il interagit avec le module de détection et celui d'acquisition des données afin de d'obtenir des informations complémentaires. Le mécanisme de diagnostic sera détaillé plus tard.

4. **Opérateur:** l'opérateur lui-même constitue un système de surveillance, ayant un rôle plus ou moins important; les résultats du diagnostic passent par lui. La composante opérateur est prévue pour les cas où certaines composantes de la surveillance ne sont pas automatisées; elle permet et guide l'intervention manuelle de l'opérateur.

1.2.2 Mécanismes de surveillance

Deux cas se présentent: le premier est lié aux mécanismes de détection de fautes ou de prédiction d'erreurs, basé sur des vérifications périodiques à spécifier; le second correspond à des situations où une faute est détectée et doit être immédiatement traitée. Il faut donc prévoir des mécanismes d'association *exception-traitement.* Pour cela nous avons prévu un module de coordination qui aura pour fonction d'identifier les opérations à effectuer à chaque séquence.

Le module de séquencement des opérations

Le niveau coordination, dirigé par un module **Simulator**, gère les séquences d'opérations assurées au niveau de l'application simulée à l'aide de sous-modules indépendants. Ceci se traduit par la modélisation de la figure 1.11, où la coordination de trois opérations séquentielles est spécifiée.

Séquence 1: Afin de simuler un procédé fautif, MSS active les capteurs d'interface afin d'introduire une erreur dans l'application simulée.

Séquence 2: une deuxième séquence vise à laisser la faute se propager. Nous simulons ainsi la propagation des fautes conformément à la réalité.

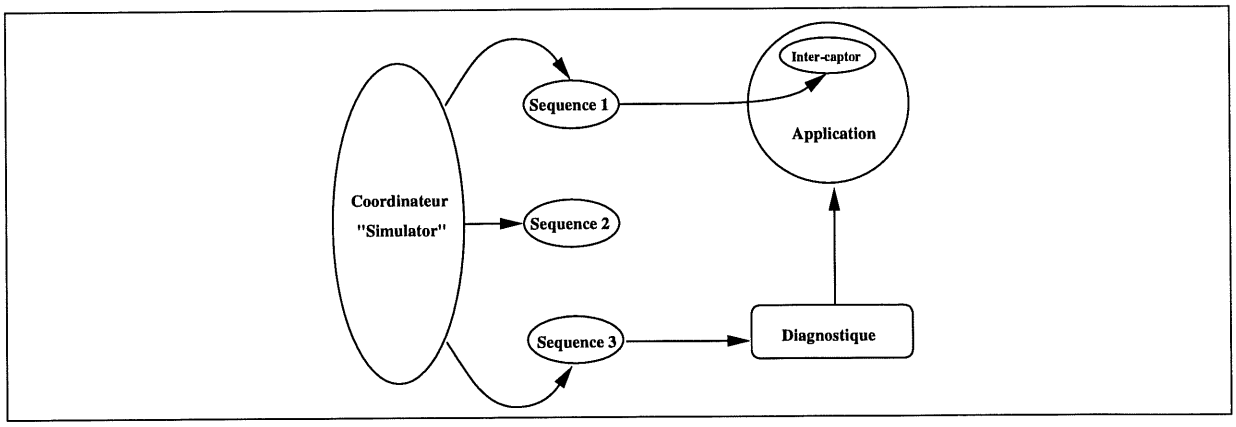

Figure 1.11 – *Coordination des opérations dans MSS*

Séquence 3: à cette étape l'application requiert un diagnostic.

Un autre avantage de ce mécanisme est lié à la configuration des séquences d'opérations. Si on décide d'accélérer le déroulement des opérations, MSS offre la possibilité de fixer la période de vérification, ceci revient à fixer la durée en secondes, nécessaire à chacune des trois séquences.

Mécanismes de détection

La détection d'erreurs et celle des fautes forment deux processus de nature différente.

La détection des fautes: c'est le moyen de prédire l'occurrence d'erreur par effet de propagation, soit latéralement entre modules d'un même niveau (dans le cas d'une application), soit verticalement entre modules de niveaux différents (entre objets). Un module de détection de fautes est conçu dans le but de communiquer au module de diagnostic les informations nécessaires pour pouvoir déterminer les origines des fautes. Une simple comparaison de la variable de contrôle surveillée avec sa correspondante idéale permet de déterminer l'état de l'objet considéré. L'état de l'objet est représenté par une variable d'état **Signe** qui figure dans l'objet surveillant. Celle ci peut prendre trois états

différents:

- plus que prévu (+)
- moins que prévu (-)
- comme prévu (0)

La détection des erreurs: c'est un processus non périodique. C'est-à dire qu'à chaque occurrence d'une faute, une interruption est déclenchée et le module de détection de fautes est activé. Chaque objet possède son propre module de détection d'erreur, qui contrôle un certain nombre de variables d'état jugées critiques.

Mécanismes de diagnostic

La détection des anomalies dans les processus continus occupe une place de plus en plus importante dans la conception des systèmes industriels. Ce besoin a conduit les chercheurs à concevoir et développer une variété de techniques de diagnostic temporisées et interactives dans différents domaines [15]. Parmi celles-ci on note:

- Les techniques par synthèse d'arbres.
- Les techniques des graphes signés.
- Les systèmes experts de diverses sortes.

À la conception de MSS, nous avons prévu une bibliothèque d'algorithmes dont l'opérateur dispose afin de surveiller son application, déjà construite. Le moyen qu'offre MSS pour effectuer un choix est décrit dans le chapitre Interface graphique. Dans la présente section, nous décrivons une de ces techniques dans le but d'illustrer le principe de fonctionnement de tels algorithmes.

Les graphes signés: il est évident que les caractéristiques de propagation des fautes dans un système sont basées principalement sur l'évolution des variables d'état dans chacune de ses composantes [8]. Il est cependant possible d'identifier la première cause d'une anomalie de fonctionnement donnée, pourvu que les variables d'état ainsi que leurs dépendances soient facilement identifiables. L'idée est de tracer un graphe, dérivé systématiquement à partir des équations caractérisant l'application considérée. Ceci a amené H. Matsuyama dans [1], à classifier les variables d'état suivant leurs caractéristiques mathématiques.

Nous allons expliquer brièvement le principe de fonctionnement d'une telle technique, basée sur la théorie des graphes. Pour cela, nous allons utiliser un exemple d'application, "Buffer", figure 1.12

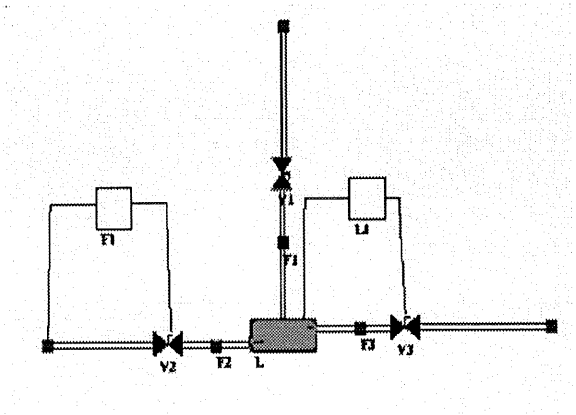

Figure 1.12 – *L'application "Buffer tank"*

F1, F2, F3, L, V1, V2 et V3 représentent les variables d'états caractérisant chacun des acteurs présents dans l'application, d'un point de vue mathématique. Cette application est représentée par le graphe de la figure 1.13.

Dans ce graphe, chaque noeud représente une variable d'état et chaque arc représente l'influence immédiate entre deux noeuds.

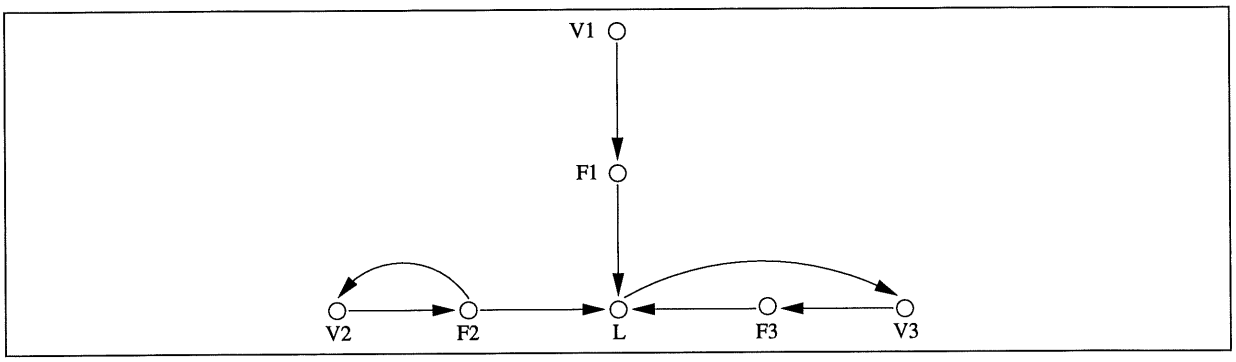

Figure 1.13 – *Le graphe signé de l'application "buffer tank"*

Les noeuds peuvent avoir trois états qui sont représentés par les signes suivants:

1. Haut: (+) plus que prévu.
2. Bas: (-) moins que prévu.
3. Normal:(0) comme prévu.

Si un noeud présente un signe différent de (0), le système est fautif.

Dans le cas des noeuds contrôlés, comme dans le cas du noeud L associé au réservoir, nous avons supposé que les signes possibles sont les mêmes, en signalant que le signe observé est (0) sauf que ça pourrait être (-) ou (+) si le noeud n'était pas contrôlé.

Les branches aussi possèdent des signes qui dépendent essentiellement de la nature de l'effet d'un noeud sur un autre. Les signes possibles qu'une branche peut avoir sont:

1. (+) effet positif; e.g. l'augmentation de la pression dans un réservoir fait croître le débit à la sortie de celui-ci.
2. (-) effet négatif; e.g. l'ouverture d'une valve diminue la pression à l'entrée de celle-ci.

Un noeud dont le signe est inconnu en recevra un au cours de l'exécution de l'algorithme. Cette opération d'affectation suit plusieurs règles [8].

Definition 1 *Un noeud n_j est dit valide si $\Psi(n_j) \neq 0$*
où $\Psi(n_j)$ est le signe de n_j

Definition 2 *Une branche b_k est dite consistante si elle satisfait aux conditions suivantes:*

1. $\Psi(\sigma^+ b_k) = \pm$, $\Psi(\sigma^- b_k) = \pm$ *and* $\Psi(\sigma^+ b_k)\Psi(b_k)\Psi(\sigma^- b_k) = +$
2. $\Psi(\sigma^- b_k) = +$ *and* $\Psi(\sigma^+ b_k)\Psi(b_k) = +$
3. $\Psi(\sigma^- b_k) = -$ *and* $\Psi(\sigma^+ b_k)\Psi(b_k) = -$
4. $\Psi(\sigma^+ b_k) = +$ *and* $\Psi(\sigma^- b_k)\Psi(b_k) = +$
5. $\Psi(\sigma^+ b_k) = -$ *and* $\Psi(\sigma^- b_k)\Psi(b_k) = -$

òu $\Psi(b_k)$ est le signe de la branche b_k, et
$\Psi(\sigma^+)$ et $\Psi(\sigma^-)$ sont respectivement les signes des noeuds initial et final de la branche b_k.

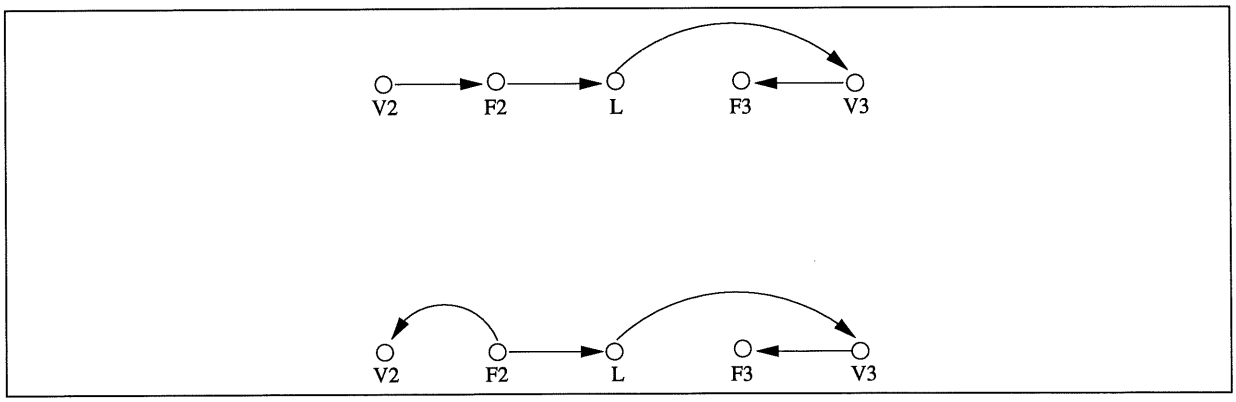

Figure 1.14 – *Les deux composantes connexes maximales*

Definition 3 *Un patron est une combinaison de signes affectée à un arbre.*

En utilisant ces définitions, la recherche de l'origine d'une faute se réduit à celle des composantes connexes maximales.

Appliquée à notre exemple, la technique des graphes signés produit les possibilités présentées dans la figure 1.14.

Les grandes lignes de l'algorithme: identifier l'origine de faute, comme dans l'exemple 1.13, nécessite les étapes suivantes.

1. attribuer un signe à chacun des noeuds dont le signe est indéterminé.
2. énumérer les graphes obtenus en appliquant les différents patrons.
3. modifier les graphes obtenus en éliminant les noeuds de signe (0) et les branches inconsistantes.
4. tester si chacun de ces graphes est bien un arbre à racine unique.
5. déduire que chaque racine d'un arbre est une cause possible des fautes.

Ceci décrit brièvement la technique des graphes signés. Cet algorithme a été élaboré et est détaillé dans [9]. Nous l'avons implanté dans le cadre du développement du système MSS, qui constitue dans sa version actuelle, un système prêt à être enrichi par ces différentes techniques de surveillance qui renferment aussi bien les algorithmes de diagnostic que ceux de prédiction.

Le terme "enrichissement" du système vient du fait que MSS est fondé sur la propriété d'extensibilité, ce qui demande des accessoires et des outils, qui permettent de faciliter l'exploitation du système.

Nous allons décrire dans le chapitre suivant les caractéristiques principales de l'interface graphique, qui consiste en un système indépendant qui communique avec la partie motrice de MSS.

CHAPITRE 2

L'interface graphique

"Software stands between the user and the machine".
Harland D. Mills

Concevoir une interface graphique implique aussi bien de résoudre des problèmes d'ergonomie, de modèles de représentation, que de méthodologies de conception et d'implantation. L'aspect extérieur d'un logiciel est ce que l'utilisateur perçoit directement du système et qui conditionne son opinion aussi bien par le confort d'utilisation que par la facilité d'apprentissage et, par la suite, détermine l'acceptation du logiciel par l'utilisateur [4]. En effet, il faut tenir compte de l'utilisateur dans les différents stades de la conception et d'implantation.

Pour réussir une bonne interface graphique, il faudrait définir précisément les qualités attendues des interfaces utilisateur telle que la visibilité, la transparence, la prévisibilité, la concision, la bonne représentation de l'écran.

La mise en oeuvre de chaque étape de la méthodologie de conception de l'interface utilisée peut poser des problèmes pouvant nuire à la qualité de l'interface finalement obtenue. Notre objectif dans ce chapitre est de nous focaliser sur l'étape de réalisation de notre interface, en étudiant les classes d'outils graphiques mise à notre disposition. Pour cela, nous allons procéder à la présentation de quelques généralités liées à la conception des interfaces graphiques, ensuite nous à détaillons l'architecture du système MSS, avant de présenter les différentes fonctionnalités de ce dernier.

2.1 Problématique de la conception d'interfaces Graphiques

Dans ce domaine, les progrès constants du matériel et des logiciels font que les systèmes informatiques actuels évoluent vers une plus grande interaction. Parallèlement à cette évolution, les applications s'adressent de plus en plus à des utilisateurs pour lesquels une bonne acceptation du système passe par la qualité de son interface. Cependant, il est fréquent que les programmeurs d'applications graphiques, réalisent leurs interfaces à l'aide de commandes ou langages de bas niveau. Il en découle alors des applications où le code destiné à l'interface se trouve mélangé au code propre au besoin de l'application. Ce qui pose les problèmes suivants:

- l'impossibilité de réutilisation des composants de l'interface.
- difficulté du prototypage rapide de l'application visée.
- risque d'aboutir à des interfaces mal conçues, puisque chaque programmeur utilise ses propres techniques d'interaction et de présentation.

Depuis plusieurs années, de nombreux efforts ont été consacrés à la réalisation de systèmes visant à décharger les applications et les programmeurs de certains problèmes liés à l'interface utilisateur.

Tenant compte de l'aspect temps réel lié au contrôle, ainsi que du matériel mis à notre disposition, nous avons opté pour l'utilisation d'un outil nommé **GARNET** (**G**enerating an **A**malgam of **R**eal-time, **N**ovel **E**ditor and **T**oolkits). Celui-ci permet la création d'une interface graphique de haute qualité.

2.1.1 La boîte à outils GARNET

Garnet est un ensemble d'outils que l'on peut décrire sous deux niveaux. Le premier, nommé **GARNET Toolkit** [12], consiste en un mécanisme permettant au programmeur de coder son interface graphique. Le deuxième niveau permet aux programmeurs et non-programmeurs de générer des dessins définissant ainsi leur interface graphique.

GARNET Toolkit consiste en une bibliothèque de procédures adaptées à l'écriture d'interfaces personne-machine. L'éventail des fonctions offertes définit différents niveaux d'abstraction. Pour simplifier, la figure 2.1 les organise en deux catégories: gestion du poste de travail et gestion du dialogue.

Écrit en Common Lisp pour X window, **GARNET** possède la propriété de portabilité sous différents environnements. Il est capable de fonctionner sous n'importe quel environnement Common Lisp, incluant Allegro (Franz), Lucid, CMU, Harlequin, CLISP et AKCL, de plus il fonctionne sur différentes stations de travail comme Sun, DEC, HP, Apollo, IBM 6000, TI, SGI et NeXTs utilisant X/11. Il supporte X/11 R4 ou R4 utilisant l'interface standard CLX. Cependant Garnet n'utilise pas les objets Common Lisp

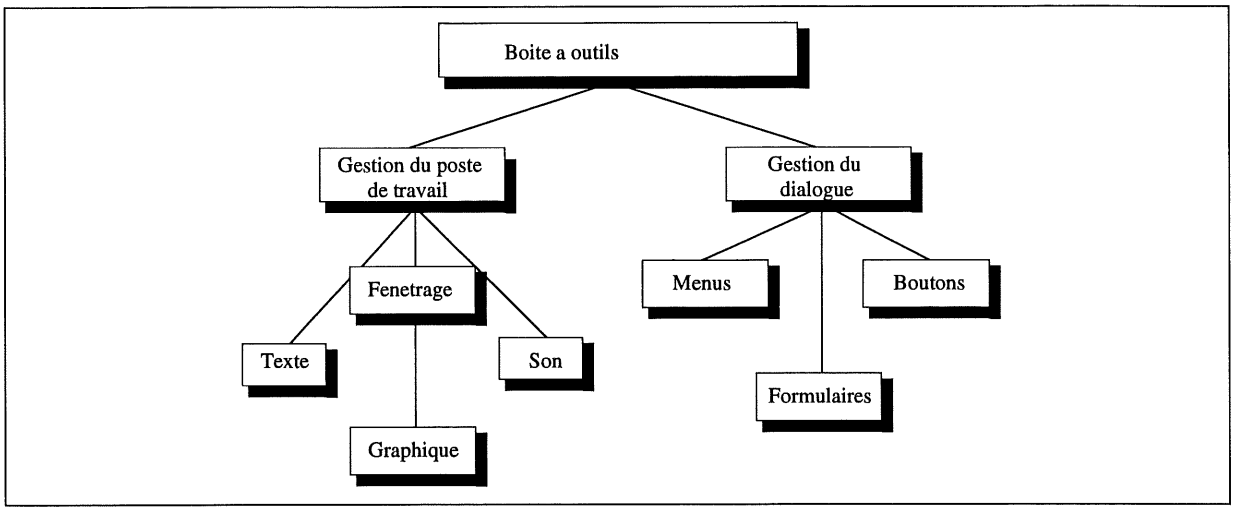

Figure 2.1 – *Les fonctions usuelles d'une boîte à outils*

(CLOS) ni tout autre outil comme Xtk ou CLIM.

GARNET est conçu pour supporter des interfaces contenant plusieurs objets graphiques que l'utilisateur peut manipuler en utilisant la souris ou le clavier. Il permet de créer des interfaces pour des applications du type:

- systèmes experts,
- éditeurs graphiques,
- éditeurs de graphe et d'arbres,
- jeux,
- simulation et diagnostic.

Avantages et inconvénients de la boîte à outils Garnet

Les boîtes à outils telles que Garnet présentent principalement quatre avantages :

1. La portabilité: elles offrent une grande portabilité au niveau des applications interactives.

2. L'extensibilité des fonctions: selon le principe des bibliothèques de fonctions, les boîtes à outils offrent la possibilité d'enrichissement par de nouvelles fonctions ou de nouveaux objets.

3. La souplesse d'utilisation: les boîtes à outils offrent au concepteur des fonctions allant du contrôle total de la machine à l'utilisation des services évolués de la boîte.

4. L'intégration de critères ergonomiques: n'ayant pas de connaissances ergonomique relatives à la présentation de l'information à l'écran, ni des compétences particulières dans le domaine de la psychologie cognitive, le programmeur dispose d'un éventail d'objets de présentation répondant au minimum de critères ergonomiques, sans pour autant offrir une méthode ergonomique qui guide le programmeur dans la réalisation de l'interface [14].

Les inconvénients de la boîte à outils Garnet sont nombreux. En voici quelques uns :

1. Mauvaise décomposition modulaire qui remet en cause l'aspect itératif de la mise au point du système interactif.

2. Duplication d'efforts: un style de programmation identique pour plusieurs applications.

3. Spécificité de l'interface utilisateur vis à vis des applications de l'utilisateur.

À ce stade nous avons étudié les ressources matérielle et logicielle dont nous disposons. Nous allons passer aux détails des différentes vues de MSS, en se basant sur les fonctions principales de notre système ainsi que les différentes règles liées au développement des interfaces graphiques. Pour cela, nous commençons par définir un squelette ou une structure générale du système, pour passer ensuite à appliquer certaines règles qui nous ont permis de réaliser notre système graphique.

2.2 Réalisation du squelette du système MSS

Les principes directeurs à la conception des squelettes d'applications sont en réalité mal définis. Il y a plusieurs raisons à cela dont le récent développement des boîtes à outil mais surtout l'absence de modèles d'architecture éprouvés [14].

L'objectif d'un squelette d'application est de fournir une structure logicielle réutilisable qui assure une grande partie des fonctions de l'interaction avec l'utilisateur. Rappelons que les fonctions de l'application doivent répondre aux besoins conceptuels de la tâche et que l'application doit avoir la possibilité de demander à l'utilisateur d'intervenir lorsque son fonctionnement l'exige.

Dans la suite nous allons nous intéresser à la conception d'une vue principale de MSS.

2.2.1 Architecture du système MSS

La figure 2.2 décrit l'architecture du système.

Au sommet de la hiérarchie, nous retrouvons le contrôleur qui remplit ses fonctions de coordination. Celles-ci seront détaillées au cours de ce chapitre.

Les objets composants se répartissent en deux catégories: ceux qui reproduisent des objets du monde réel en rapport étroit avec les fonctions du système tels le "Tank", la "Valve", etc, et ceux d'utilité générale tels l'aide et les menus. Nous ne reviendrons pas sur le fonctionnement des objets généraux dont on trouve des réalisations prêtes à l'emploi dans GARNET.

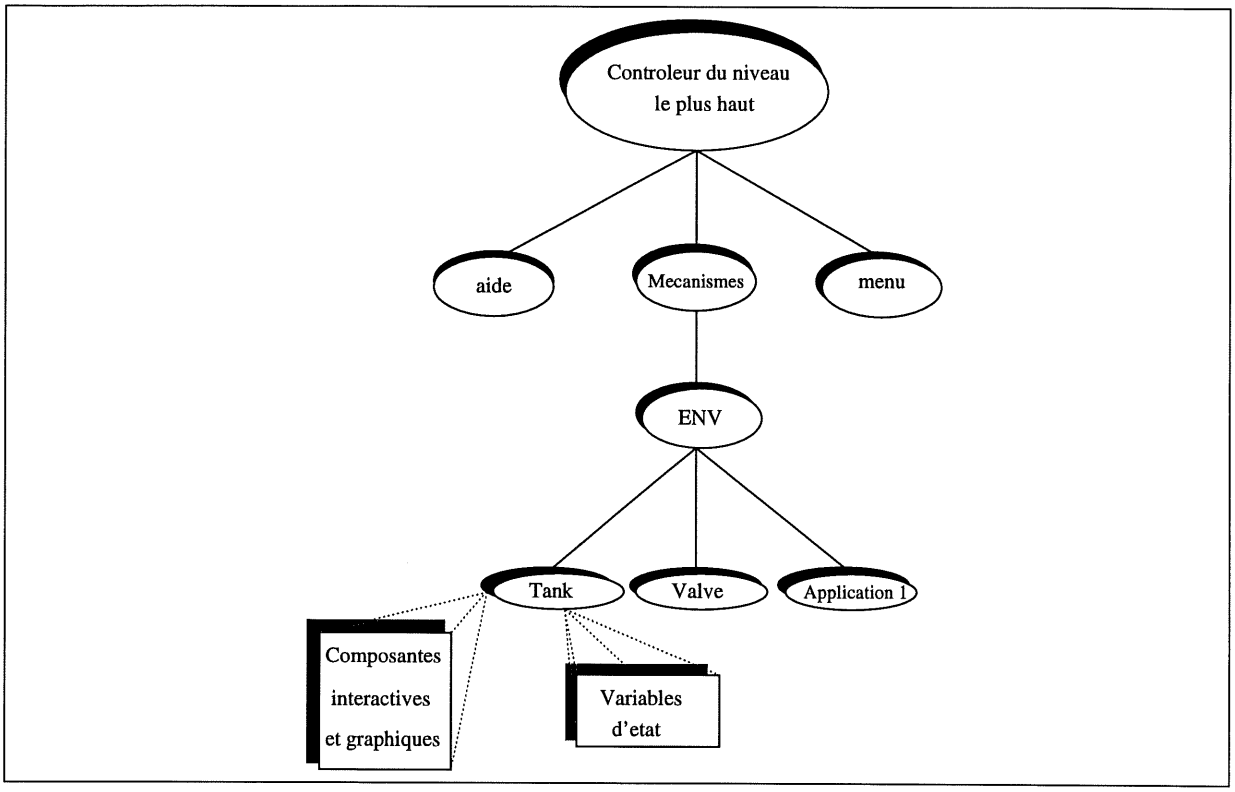

Figure 2.2 – *Architecture du système MSS*

La réalisation des objets élémentaires de la première catégorie respectent le modèle MSS décrit au chapitre 1. Par exemple pour l'objet **Tank**:

- la présentation sait dessiner une forme d'un réservoir à une certaine localisation sur une surface de restitution. Selon le point ou le bouton désigné avec la souris, elle interprète les actions de l'utilisateur soit comme un déplacement, soit comme une pression sur un interrupteur. L'interprétation de ces actions (déplacement ou pression) est signalée au module contrôleur de l'objet qui est représenté par les composantes interactives.
- le contrôleur traduit une pression de l'interrupteur par un appel d'une fonction pré-définie. Ce dernier traduit aussi la valeur d'état *Level* en une hauteur. Cette hauteur est utilisée par la présentation pour dessiner et animer l'objet en question.

Comme nous l'avons mentionné au chapitre 1, les objets "Tank" et "Valve" tout comme pour les autres, sont conçus selon le même modèle. Celui-ci peut faire l'objet d'une réalisation sous forme de squelette réutilisable.

Chaque objet élémentaire tend à reproduire le comportement intrinsèque d'un objet physique. Tout comportement spécifique reste néanmoins sous l'influence des contraintes provenant de l'environnement. Une application dans MSS voit cohabiter une valve, un tank, et plusieurs autres composants. Afin que cette cohabitation respecte les lois du monde physique, un agent doit la faire respecter. L'agent **ENV** de la figure 2.2 remplit cette fonction d'arbitre. Il n'a pas de présentation intrinsèque à proposer.

Aprés avoir considéré l'architecture générale du système, nous pouvons aborder les détails de réalisation: le niveau des échanges entre l'application et l'interface, le dialogue à plusieurs fils d'activités et la présentation des objets.

2.2.2 Les fonctions du système MSS et son image

La figure 2.3 donne un aperçu de l'image du système MSS. Le principal objectif de cette image est d'être fidèle le plus possible au monde réel.

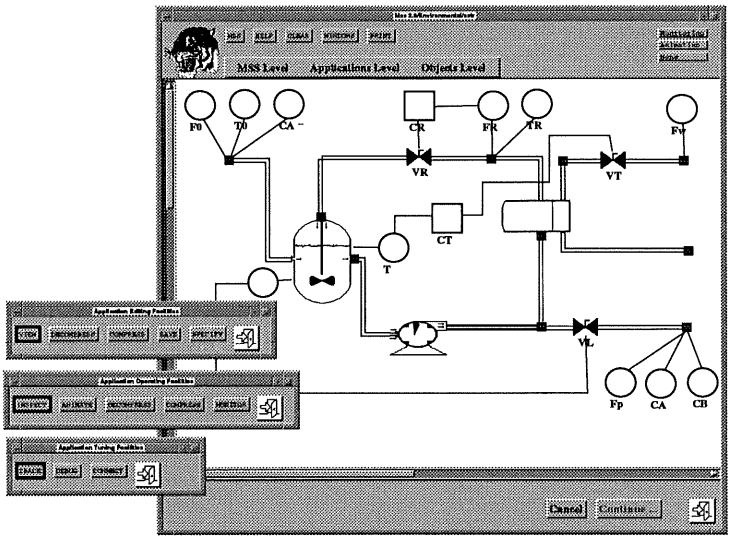

Figure 2.3 – *Image du système MSS*

Puisqu'il s'agit d'informatiser une activité du monde réel, l'image doit respecter les éléments marquants de celui-ci. En ce qui concerne les environnements créés lors du développement de MSS, les objets ont été identifiés à partir de documents spécialisés dans ce domaine. Nous avons en particulier relevé le rôle central ainsi que l'apparence physique associée à chaque objet.

L'application présentée par la figure 2.3 est prise du domaine des traitements des eaux. Les éléments graphiques sont les reproductions des procédés utilisés dans le monde physique. Précisons que leur structure interne a été présentée dans le chapitre précédent. Nous nous intéressons ici à leur apparence ainsi qu'à leur capacité de communiquer avec

l'utilisateur. En ce qui concerne l'apparence, elle consiste en une ou plusieurs images (bitmap) associées à des objets élémentaires (e.g. rectangle). La figure 2.4 présente un (CSTR) ou tout simplement un réservoir d'eau, possédant une hélice, pour agiter l'eau. Les deux fenêtres des deux cotés de l'objet représentent deux résultats de la réaction de l'objet à une action de la souris. La fenêtre à gauche résulte de l'inspection de l'objet. Celle-ci permet aussi de générer différents instruments de mesure. Celle à droite résulte de la fonction d'aide instaurée dans l'objet lui-même. Elle permet de donner des explications sur les causes et les effets de la faute s'il y a lieu.

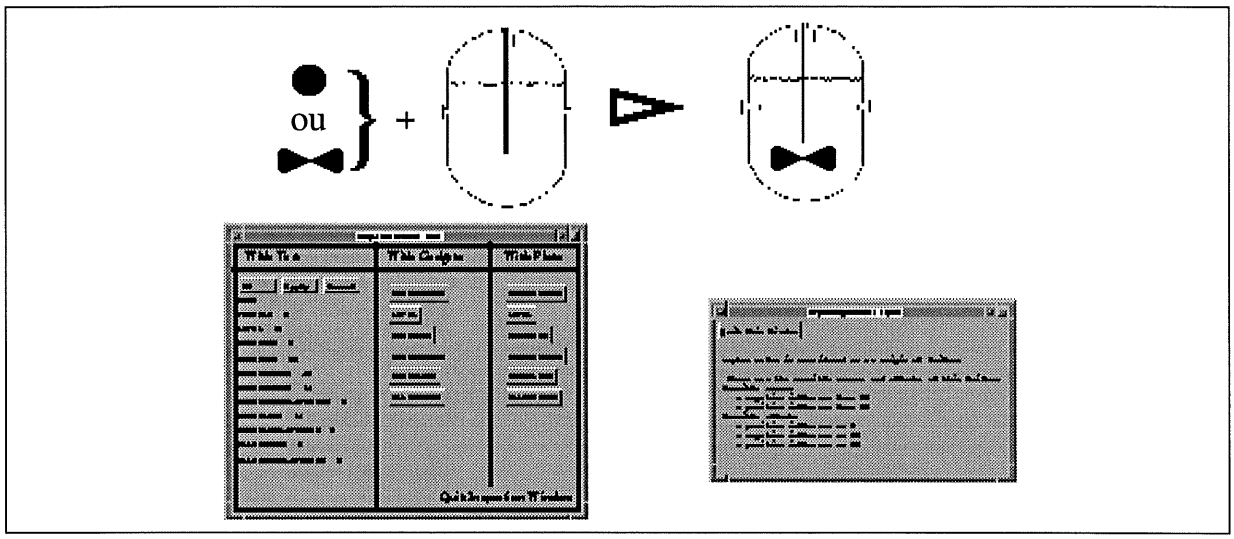

Figure 2.4 – *Représentation et interaction de l'objet*

Reproduction des procédures du monde réel

Dans le monde réel, le concepteur décide de la création d'une application et de l'installation d'accessoires matériels. Ces décisions se concrétisent par l'ajout, le retrait et le déplacement des objets graphiques sur l'écran principal. Notre système reproduit cette méthode de travail. Les objets graphiques sont réunis dans une palette concrétisée par les classes ou les familles de dispositifs du domaine. La palette est construite à partir

d'un ensemble de menus, représentés chacun par le nom de la famille d'un ensemble de dispositifs. Dans la figure 2.5, l'opérateur vient de désigner le symbole qui représente le dispositif "*Tank CSTR*" de la classe "TANKS", celui-ci apparaît sur l'écran, après que l'opérateur ait désigné son emplacement.

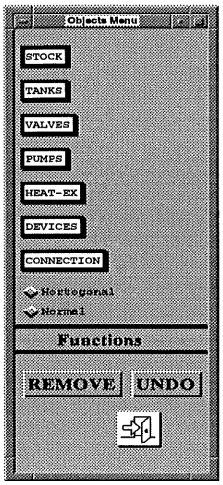

Figure 2.5 – *La palette des objets*

Noter que toute action élémentaire de l'utilisateur est précédée par un message d'aide qui apparaît en bas de la fenêtre principale, ainsi que sur la fenêtre **Guide** représentée par la figure 2.6.

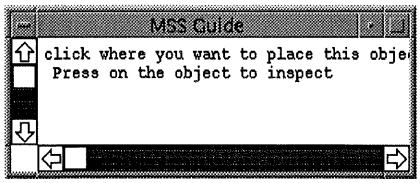

Figure 2.6 – *Enregistrement des messages d'aide.*

2.2.3 Échanges entre le système et son interface

La communication entre le système et son interface occupe une place importante en terme de concepts de celui-ci. Le module imposé par **Garnet** rendait possible ce niveau d'échanges.

Dans MSS, une application ou un objet utilise la valeur associée à la "Température" mais doit tout ignorer des techniques de présentation; par exemple, afficher une température ne relève pas du modèle mathématique qui rend compte des phénomènes thermiques. De même, le thermomètre est un objet qui peut représenter d'autres valeurs entières que celle de la température. La technique de réalisation des indirections n'est pas unique; table de liaison, valeurs actives à la Loops, système de règles. Le choix dépend essentiellement des outils de réalisation [4]. Ces indirections sont principalement gérées par Garnet, sauf pour les cas où les objets de contrôle sont fabriqués par MSS, comme l'inspection par des courbes, ou des capteurs. Dans ces cas, la technique des valeurs actives à la Loops convient parfaitement. Un exemple d'échanges est illustré dans la figure 2.7.

Chaque objet de présentation est construit à partir d'une structure graphique et un module identique à celui des animateurs détaillés au chapitre précédent. Le rôle d'un animateur est de surveiller en permanence une des variables d'état et changer d'apparence chaque fois que celle-ci est modifiée. Dans ce cas, l'échange est effectué du système vers l'interface. L'inverse est aussi possible vu qu'un changement au niveau interface peut influencer le système. Par exemple un changement de la valeur "Level" sur le panneau d'inspection qui est l'objet de présentation, entraîne une rectification de la variable d'état correspondante dans l'application. Celle-ci est traduite par un changement d'apparence de l'application en question.

En résumé, quelle que soit la nature du signal entre l'application et le contrôle du niveau

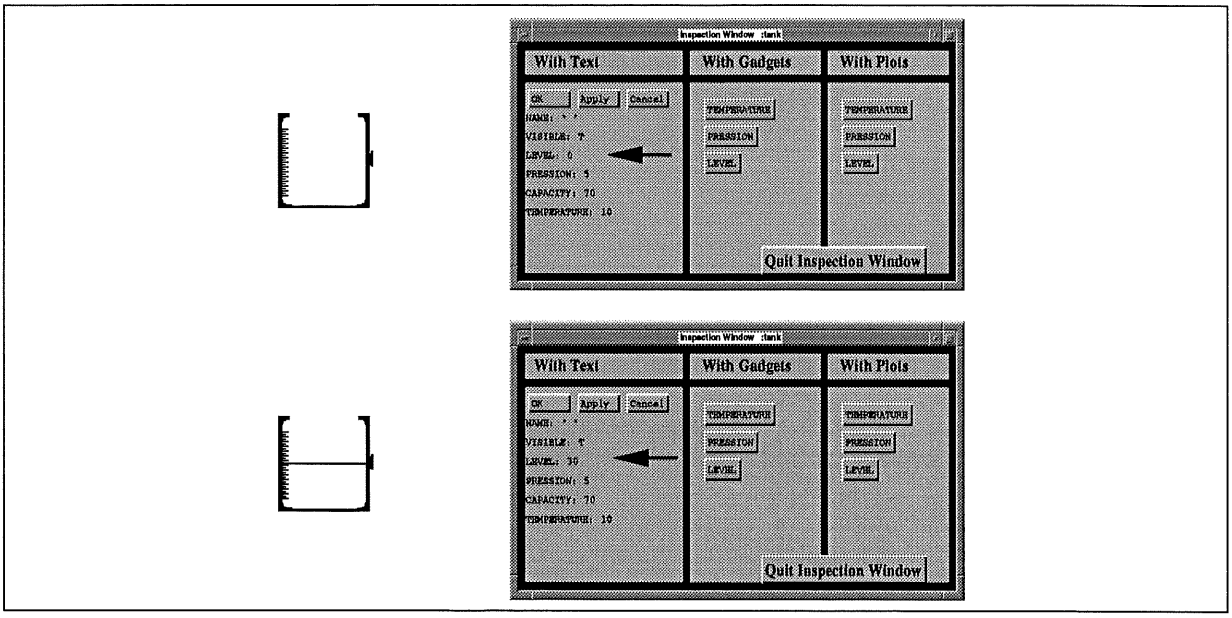

Figure 2.7 – *Échange entre l'application et son interface*

le plus haut, le sujet doit être toujours un concept ou une représentation de l'application, mais jamais un élément de présentation. Il est évident que ce genre d'échanges, lié à l'animation graphique, doit prendre en compte plusieurs autre facteurs, comme l'allocation des ressources. Les ressources envisagées ici sont l'unité centrale et l'écran. Cette tâche de nature complexe est assurée par l'outil de réalisation.

2.2.4 Dialogue à plusieurs fils d'activités

Au cours du dialogue entre le système et l'utilisateur, il arrive que ce dernier fasse appel à un service qui à son tour, a besoin de l'intervention de l'utilisateur pour poursuivre sa tâche. Par exemple, la figure 2.8 montre une image du système MSS après une demande de renseignements sur les causes et les effets de l'erreur présente. Le service d'aide présente quelques informations préliminaires et attend l'intervention de l'utilisateur pour compléter sa fonction. Rien n'impose à l'utilisateur de répondre immédiatement. Celui-ci

peut poursuivre la manipulation d'autres objets avant de revenir au service d'aide.

Par contre le bouton "Continue" en bas de la fenêtre, clignotera jusqu'à l'intervention de l'utilisateur pour pouvoir continuer le processus d'animation et de surveillance.

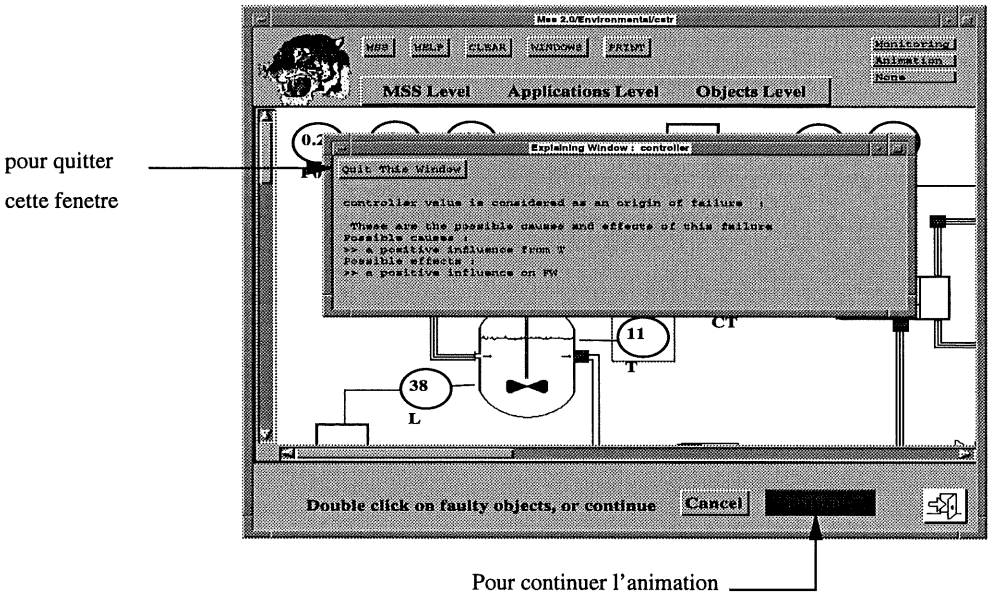

Figure 2.8 – *Dialogue à plusieurs fils d'activités*

Nous avons ainsi construit un squelette du système qui intègre un ensemble de services ergonomiques, tels que l'aide, la gestion des erreurs, les fonctions d'historique et les explications. Il inclut aussi des objets graphiques et des fonctions générales pré-définie qui renforcent la cohérence, l'uniformité ainsi que la simplicité du système.

Nous allons passer ensuite aux détails des fonctions les plus importantes dans MSS, à savoir le système des alarmes et d'aide à la conduite ainsi que la démarche prise pour faciliter les interventions de l'opérateur.

2.3 Les méthodes de présentation de l'information

Le système d'interface de MSS comprend plusieurs écrans de visualisation, qui représentent un ensemble de vues constituées de plusieurs images. Chacune de ces vues présente un certain nombre d'informations. Les vues peuvent être composées de deux parties; une statique et l'autre dynamique. La première est non modifiable et reste en permanence à l'écran, tandis que la seconde est constituée d'informations animées. Un exemple de ce genre de vues est présenté à la figure 2.9. Cette vue est à l'origine de la fonction "inspect" qui permet de fournir des informations concernant les variables d'état de l'objet inspecté.

Parmi les informations dynamiques utilisées dans MSS nous pouvons citer:

- Le symbole: il représente graphiquement un composant physique, par exemple une vanne, un niveau d'eau; ainsi il peut apparaître et disparaître à l'écran et selon les besoins de l'application il peut également se déplacer dans une zone pré définie ou changer de couleurs.
- Le message: il peut être utilisé pour avertir l'opérateur de l'apparition d'une alarme, ou pour le conseiller dans une action de correction d'une variable; les modes d'affichage des caractères constituant le message, la couleur, la taille des caractères, peuvent être modifiés suivant la gravité de la situation.
- Le cadran: celui-ci correspond graphiquement à un indicateur fléché mobile, animé suivant la valeur d'une variable. L'indication de la valeur courante est donnée entre un seuil minimal et un seuil maximal.
- Le compteur: il fournit la valeur numérique correspondant à la variable du processus qui lui est associé.
- La courbe: elle permet de synthétiser l'historique d'une variable.

Chaque information animée peut être associée à des informations statiques, rappelant son contexte d'utilisation et sa signification. Par exemple, pour une fonction de type courbe,

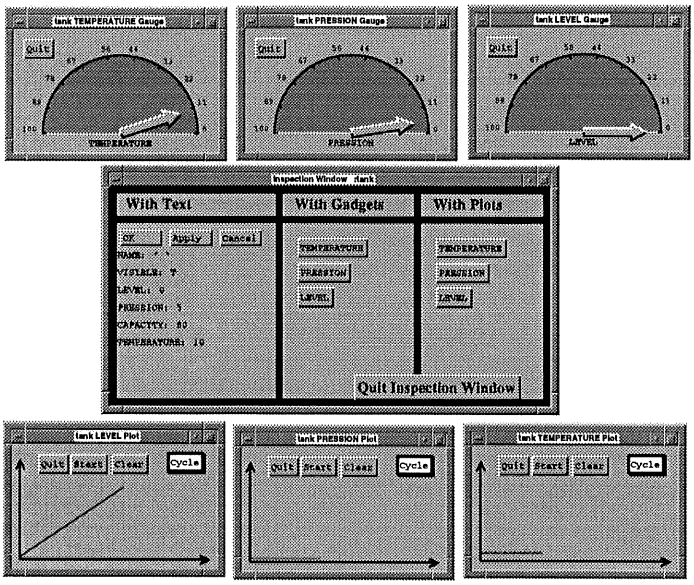

Figure 2.9 – *Une vue d'inspection*

il faut distinguer d'une part l'information graphique dynamique permettant l'affichage de points à l'écran en temps réel, et d'autre part, les informations graphiques statiques constituant son environnement, telles que le nom de la variable, les axes, les graduations, les unités, etc. Une fois définies, ces méthodes de présentation sont mises en oeuvre afin de construire les accessoires et les propriétés qui enrichissent notre système.

2.3.1 Les alarmes

Parmi les aides techniques qui peuvent être fournies à l'opérateur dans les salles de contrôle, les plus anciennes et les plus usitées sont les alarmes. Elles peuvent être définies comme des dispositifs prévus par le concepteur pour avertir l'opérateur d'une anomalie dans le déroulement du processus. Le système d'alarme est en relation étroite avec le mécanisme de contrôle, et plus précisément, de diagnostic. Cette relation a été présentée

au chapitre précédent. Dans cette section nous nous préoccupons de l'aspect présentation des alarmes.

Dans les salles de contrôle, une alarme est souvent associée à un signal sonore plus ou moins "stressant" pour l'opérateur, allant du beep sonore à une sonnerie répétitive [11]. Dans MSS, l'alarme accompagnée d'un bip sonore est affichée dans une zone spécifique de l'écran, bien visible à l'opérateur chaque fois que le mécanisme de diagnostic est lancé. À partir d'une alarme ou de la détection d'un événement anormal, l'opérateur observe des informations visant à sélectionner, parmi les informations disponibles celles qui sont les plus significatives quant à l'état de fonctionnement du procédé. L'opérateur doit considérer ces informations non seulement lors des tâches de surveillance, mais aussi lors des tâches de résolution de problèmes. De plus, lorsque les informations utiles à présenter ne sont pas disponibles, le système est capable de reconstruire à partir d'informations de niveaux supérieurs, par exemple des variables portant sur des réactions chimiques (ceci peut être facilement déduit à partir d'un graphe de causes à effets).
La figure 2.10 montre le système d'affichage associé aux alarmes. Ces informations sont accompagnées de l'affichage d'un indicateur visuel sur tout objet jugé fautif ou origine de défaillance.

Un autre outil d'aide à la conduite de l'opérateur a été mis en oeuvre. Celui-ci consiste en un ensemble d'informations reportant sur les causes et les effets de chaque faute associée à un dispositif quelconque. Ces informations sont visualisées par un double click sur l'objet considéré.

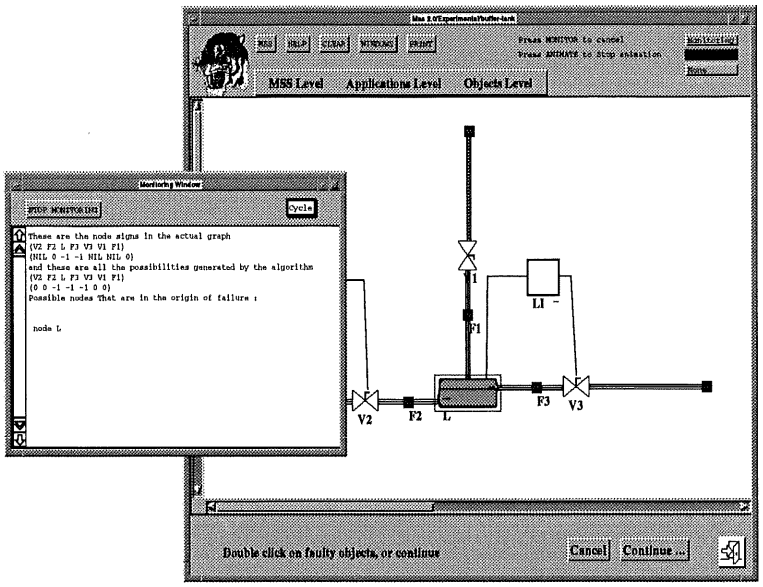

Figure 2.10 – *Fenêtre d'affichage des informations reçues du mécanisme de diagnostic*

2.3.2 Les symboles graphiques et leurs codes d'utilisation

Depuis les années 70, des organismes de normalisation internationaux se sont penchés sur le problème de la normalisation des symboles graphiques et de leurs codes d'utilisation pour les affichages du type synoptique utilisés pour le contrôle de procédés industriels.

Au cours du développement de quelques applications sur MSS, nous nous sommes basés sur les normes graphiques indiquées par les documents utilisés [5] [8]. À titre d'exemple, la figure 2.11 expose les symboles graphiques proposés par la norme ISA (particulièrement utilisée dans le secteur pétro-chimique) pour schématiser un dispositif réglant.
Pour les dispositifs les plus complexes, la représentation dans les documents est souvent très simplifiée. Nous avons eu recours à plusieurs reprises à notre propre imagination pour produire des symboles graphiques ayant une grande ressemblance avec les disposi-

tifs réels. Nous citons comme exemple le dispositif "decker" (voir figure 2.12) utilisé dans le secteur des pâtes à papier.

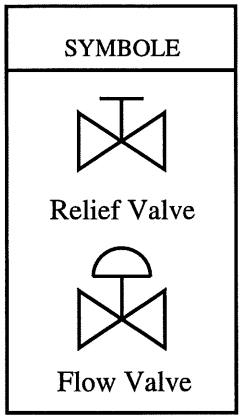

Figure 2.11 – *La normalisation des dispositifs réglant (exemple d'une valve)*

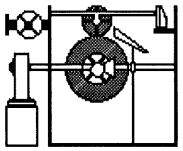

Figure 2.12 – *Représentation d'un dispositif complexe "decker" utilisé pour l'extraction des fibres de papier*

2.3.3 Organisation et structuration des informations

Une application représente le noyau autour duquel fonctionne tout le système de surveillance. L'application a été définie au chapitre précédent comme un ensemble d'objets en communication. Cependant, une application peut être elle-même un objet, c'est à dire qu'elle peut être construite à partir d'autres applications de manière à être structurée sous forme arborescente. Ce moyen est offert par la fonction "compress".

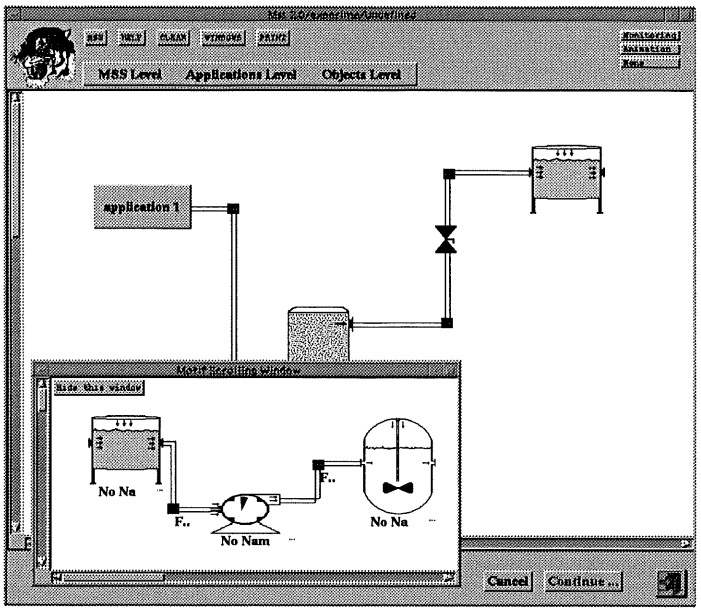

Figure 2.13 – *Visualisation d'une application contenant d'autres applications compressées.*

L'opérateur peut construire ainsi une application ou une partie de celle-ci, puis la compresser en lui donnant un nom comme tous les objets. Elle pourra être ensuite utilisée dans la construction d'autres applications plus complexes. L'application compressée, peut être visualisée par le biais de la fonction "decompress". Celle-ci permet d'afficher le contenu d'une application, qui est l'équivalent d'un objet, dans un écran indépendant (figure 2.13). Ainsi plusieurs sous-applications sont construites de cette façon, permettant un très haut degré d'optimisation du support physique de l'information.

2.3.4 Le support physique du système d'assistance

Après avoir décomposé les informations en une arborescence d'images et organiser et structurer l'information pour chacune des vues de l'interface, le concepteur de l'application est amené à choisir un moyen d'organiser ces vues.

Considérant les contraintes matérièles, MSS a été conçu suivant le principe de la centralisation de l'information sur une console unique à partir de laquelle l'opérateur à la possibilité de lancer chacune des fonctionnalités du système. Ce choix a été principalement approuvé, vu les possibilités de multi-fenêtrage offertes par l'environnement "X WINDOW". Ainsi, nous étions amené à intégrer dans le système un moyen pour gérer ces fenêtres (afficher en avant-plan, détruire ...).

2.3.5 La conformité au langage de l'opérateur

Il peut paraître évident qu'un opérateur puisse éprouver des difficultés à utiliser un système informatique dont les messages sont affichés dans une autre langue que la sienne. De telles situations peuvent apparaître lorsqu'un opérateur tend à configurer ou manipuler un processus, éprouvant des difficultés à retrouver certaines fonctions graphiques. De plus, il existe des règles de grammaire et d'orthographe qu'il faut absolument respecter pour chacun des messages provenant du système d'assistance.

Parmi les options qu'offre MSS aux opérateurs, il existe l'option de changer de langue. Les langues utilisées dans MSS sont l'anglais et le français. il y a toujours un moyen d'ajouter une autre langue. Celles-ci sont stockées dans un fichier sous forme d'un tableau de conversion.

L'ingénierie de conception et d'évaluation des interfaces personne-machine pour le contrôle de procédés industriels peut se baser sur de nombreux outils, techniques, modèles et méthodes. Notre démarche de conception et d'implantation de l'interface graphique MSS à été principalement guidée par l'outil GARNET. Quelques modèles ont été pris du système ERGO-CONCEPTOR décrit dans [11].

CHAPITRE 3

Expérimentation et extension

Nous venons de décrire MSS, essentiellement par une présentation graphique d'un processus physique ainsi que par sa décomposition en deux sous-systèmes formant son noyau. Avant de passer à un exemple d'utilisation, nous nous devons de rappeler le cadre d'utilisation de MSS.

Une première possibilité pour l'utilisation de notre système consiste à l'utiliser comme simple outil de visualisation et d'aide à la décision, sans le connecter directement au processus réel. Dans ce cas, MSS n'est couplé que de façon indirecte au processus, par l'intermédiaire de l'opérateur qui introduit les résultats de mesure dans l'ordinateur et qui agit manuellement sur le processus en fonction des résultats qui lui sont fournis. Un autre mode de fonctionnement du système qui diffère du premier, consiste à connecter directement les sorties du processus à MSS tout en laissant l'opérateur intervenir manuellement sur le processus en fonction des données qui lui sont fournies par l'ordinateur sur l'écran de visualisation.

MSS est une surcouche de procédés existants auxquels il s'adapte. Ainsi, certains de ses objets n'existent pas ou alors existent sous des formes différentes. En fonction du domaine de travail, le concepteur d'applications utilisera les objets ayant un correspondant proche dans le système cible (les objets de base) et créera le reste des objets au cours de l'étape de construction de son environnement. Dans cette optique, MSS ne doit pas perturber le concepteur vis-à-vis de son application. Au contraire, il facilite la conception détaillée en accordant d'abord une importance plus grande à la présentation de la dynamique de l'application.

Les deux exemples traités ci-dessous vont nous permettre d'accompagner la description de l'outil d'une présentation de quelques techniques de construction des applications. Il s'agit de cas simples ne demandant pas de pré-requis. Nous allons utiliser une première application tirée du domaine des pâtes et papiers afin de présenter l'expérimentation du sous-système réactif. La seconde application provient du domaine du traitement des eaux et nous permet d'illustrer le fonctionnement du mécanisme transformationnel.

3.1 Le désencrage des pâtes à papier

Le contrôle de la qualité au sein d'une installation de désencrage est aussi importante que celui de toute usine de pâte, mais il s'agit d'une activité encore plus cruciale en raison des fluctuations subies par la composition fibreuse [5].

3.1.1 Construction de l'application

La mise au point des procédés de désencrage s'articule autour de trois grands pôles: le classement des vieux papiers, le choix des produits chimiques servant à la cuisson et

au blanchiment et le choix du matériel de cuisson, de lavage, de classage, d'épuration centrifuge, de blanchiment et d'égouttage. La figure 3.1 montre une installation d'un système de désencrage. La conception d'une telle application dans MSS est basée sur les

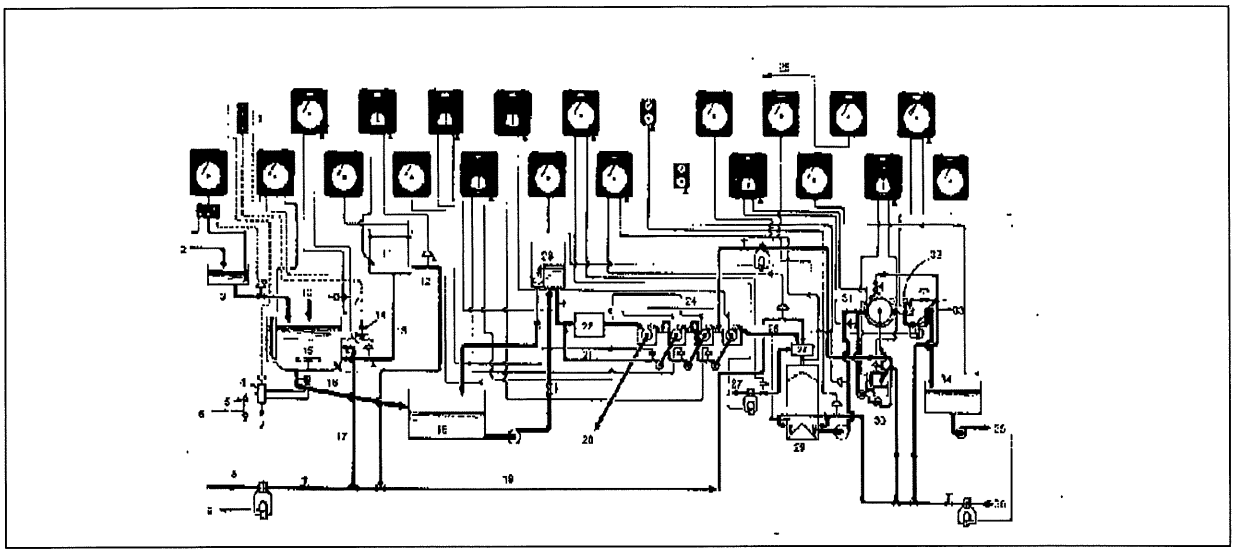

Figure 3.1 – *Procédé de désencrage classique et instrumentation connexe*

étapes suivantes:

1. Étude des différents dispositifs utilisés dans le système: forme, comportement et utilité.
2. Une fois les objets construits et rassemblés dans un environnement commun, il reste à construire l'application en les plaçant dans leurs positions finales et à les connecter par les "streams".
3. Placer les capteurs aux positions jugées critiques.
4. Placer les générateurs de fautes (ceci sera expliqué au cours de l'application suivante).
5. À l'aide de la fonction "inspect", initialiser les variables d'état de chaque objet, ceci permet de définir l'état initial du système.

6. Sauvegarder l'application.

La figure 3.2 montre le système de désencrage ainsi construit.

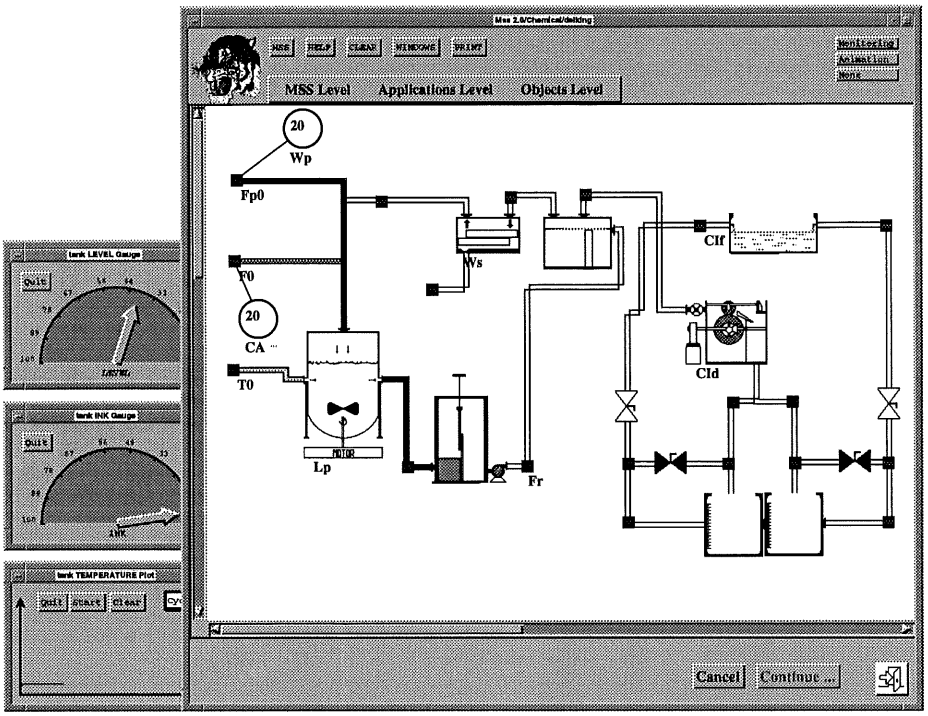

Figure 3.2 – *Procédé de désencrage construit en MSS*

3.1.2 Interaction entre les objets

Nous avons précisé que le comportement des objets est principalement défini dans le module "animateur". Dans ce qui suit, nous décrirons le déroulement et le fonctionnement du système dans le but d'illustrer le mécanisme de communication entre les objets. Cependant il faut préciser que ceci n'est qu'une description abstraite du processus vu qu'une telle activité demande des spécialistes dans le domaine. Nous avons identifié les composantes de cette application à partir de [10].

Description du procédé

Le procédé étudié suppose que le classement des vieux papiers est effectué. La pâte est extraite à partir de la source (invisible dans la figure), passe directement dans le mélangeur pour être mixée avec "l'Alkali" à une température élevée. Le choix de l'alkali dépend évidemment de la nature de la pâte et de l'encre. Le mélangeur ne fait qu'activer la réaction chimique entre l'encre et l'alkali. Les concentrations des divers produits ainsi que la température sont représentés par les variables d'état de ce dispositif. Le mélange est ensuite transféré dans un régulateur de concentration, celui-ci permet de régler le débit du mélange. La troisième étape consiste à séparer les déchets de la pâte en utilisant deux dispositifs d'épuration: épurateur 1^{er} étage et 2^{eme} étage. Le tout est pompé vers un épaississeur unilatéral pour extraire les fibres à papier. Celles-ci sont lavées en utilisant la pile laveuse qui permet de faire circuler le mélange afin de se débarrasser des déchets.

Les générateurs de fautes sont déjà installés dans l'application, leur état étant invisible car leur utilité n'intervient que dans le déroulement du système transformationnel que nous allons mettre en évidence par le deuxième exemple.

3.2 Le traitement des eaux

Le **CFSTR** (**C**ontineous **F**low **S**tirred-**T**ank **R**eactor) représente une étape très importante dans le processus d'épuration des eaux. Dans cette section, nous allons nous intéresser à la mise en oeuvre du système transformationnel. Pour cela nous allons considérer que les étapes de construction sont déjà faites, et nous passons à la description des relations entre processus pour pouvoir construire un graphe qui nous permettra de contrôler le système.

La figure 3.3 montre le système CFSTR tel qu'il est présenté dans la documentation [8].

3.2.1 Le graphe Signé du système CFSTR

Dans le but d'utiliser un algorithme de diagnostic basé sur les graphes signés au cours de l'étape de surveillance du système, il est nécessaire après la construction de l'application de créer le graphe correspondant. Celui-ci est déduit de certaines règles comme:

- Le niveau L est contrôlé par un senseur C_L. Ce contrôleur est représenté par une valve V_L que l'on ajuste à volonté. En agissant sur V_L, on agit automatiquement sur le débit F_P.
- Lorsque la pression P_P augmente, le débit F augmente aussi; par contre l'inverse n'est pas évident et reste à vérifier.
- F_P et P_T ont un effet réciproque l'un sur l'autre. Cette relation est vérifiée. Lorsque le débit F_P augmente, ceci implique que la valve V_L est entrouverte d'où moins de pression exercée en P_T, d'où P_T diminue. Par contre, P_T augmente implique F_P augmente.

Ainsi, à partir de ces règles, un graphe de causes à effets est construit (figure 1.13) et il nous faut placer les générateurs de fautes à des positions critiques c.a.d sur des variables d'état qui vont nous permettre la propagation de la faute à différents degrés de "profondeur".

Ainsi, l'étape de construction de l'application est achevée. Pour tester son déroulement, il suffit de lancer le mécanisme de simulation et de contrôle.

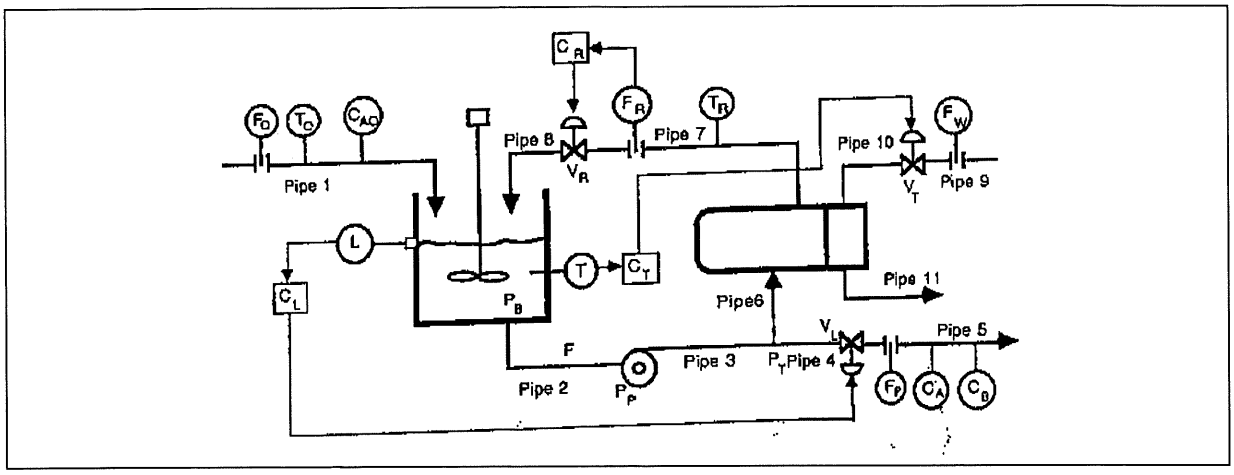

Figure 3.3 – *L'application CFSTR*

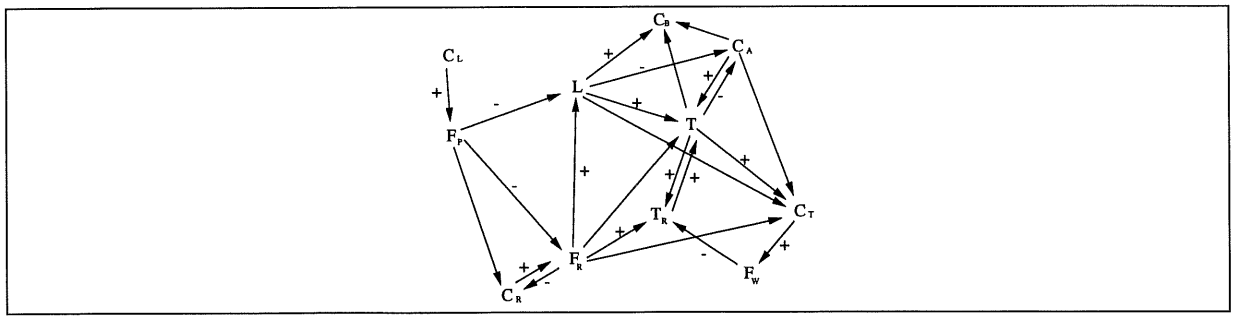

Figure 3.4 – *Le graphe signé du système CFSTR*

3.2.2 Le cycle du diagnostic

Pour maintenir le rendement d'élimination des substances chimiques dans une installation telle que le système CFSTR, il est nécessaire de surveiller continuellement la teneur en ces substances dans toutes les composantes du système. Ceci revient à placer des capteurs sur toutes les variables d'état représentant ces entités.

Une fois les composantes de l'application CFSTR animées, le module de séquencement des évènements se met à fonctionner lorsque l'opérateur active la fonction de contrôle. Les séquences de déroulement du procédé ont été expliquées dans le chapitre précédent. Nous allons nous intéresser à la séquence du diagnostic pendant la simulation du système CFSTR. La figure 3.5 montre les résultats affichés par MSS après l'opération de diagnostic.

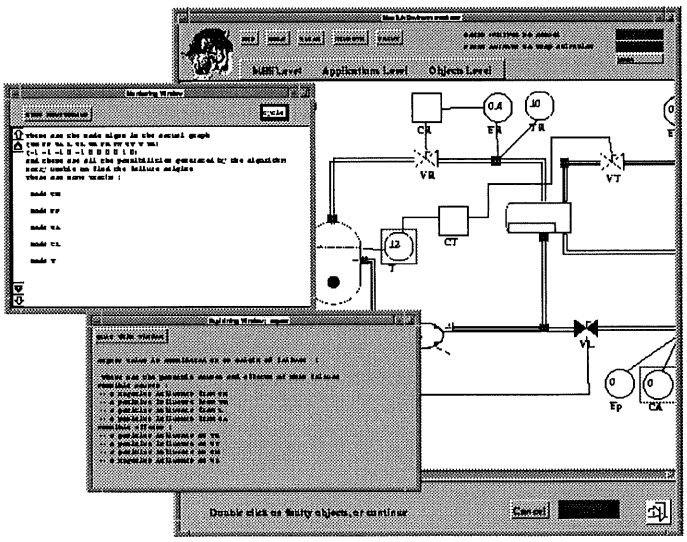

Figure 3.5 – *Les résultats du diagnostic du système CFSTR*

Une faute est détectée et le module de diagnostic est lancé, l'alarme est apparue sous

forme d'un "beep" sonore et d'un clignotement d'une lampe indiquant que le système est arrêté. Les origines possibles de la faute sont affichées sur un écran de visualisation sous forme de texte. Les dispositifs correspondants sont indiqués par un cadre qui apparaît autour de l'objet.

Il est toujours nécessaire pour l'opérateur d'avoir une explication de la situation autre que celle définie par les symboles qui apparaissent sur l'écran de visualisation. Une pression sur un objet encadré à l'aide de la souris, permet de visualiser une explication sur les causes et les effets du dispositif considéré, vis-à-vis de son entourage. Ces informations sont représentées dans la fenêtre qui est placée au dessous des résultats du diagnostic dans la figure 3.5.

3.3 Les extensions futures

Les développements futurs de ce travail consisteraient à réaliser des modules supplémentaires permettant de créer des fonctionnalités additionnelles au système. Ces modules se répartissent en trois catégories suivant leurs utilités:

1. Un ajout au niveau des facilités d'exploitation.
2. Un ajout au niveau des facilités d'édition.
3. Un ajout au niveau des facilités de mise au point.

Ces trois types d'ajouts nous permettront de faciliter la manoeuvre de l'opérateur sur les trois niveaux de MSS.

- Le niveau environnement:

    1. Facilité "Design": c'est le moyen de construire un nouvel environnement sans avoir recours à une méthode textuelle ce qui est le cas pour la version actuelle

de MSS. Il est évident que ce genre de fonctionnalités intervient dans les facilités d'édition.

2. Facilité "View": vu qu'un environnement possède aussi une apparence graphique, il est préférable de pouvoir le visualiser au cours de sa construction.

3. Facilité "Generate": ceci consiste à créer une fonction qui permet de générer tout un système spécialisé dans un domaine donné. Cette méthode nous permettra de créer une image de MSS comportant deux niveaux: application et objet. Cette image pourra être installée et configurée séparément.

- Le niveau application:

1. Facilité "Specify": pour initialiser l'application construite, sans avoir recours à la fonction "inspect". Ce processus s'étend jusqu'à la spécification et le choix de l'outil qui sera utilisé dans le mécanisme de surveillance. Cette fonction appartient à l'ensemble des facilités d'exploitation.
2. Facilité "Connect": c'est le moyen prévu pour la mise en oeuvre du deuxième mode de fonctionnement de l'outil. Ce mode consiste à lier les dispositifs ou les composantes réelles du système à leurs correspondants dans MSS. Ceci requiert quelques initialisations au niveau logiciel. Une telle facilité doit appartenir à l'ensemble des facilités de mise au point.
3. Facilité "Trace": cette fonction représentera un moyen efficace pour pouvoir suivre le déroulement du processus lors de l'apparition d'une faute. Elle nous permettra d'analyser les étapes de propagation de la faute. D'après son utilité, cette fonction est placée parmi les facilités de mise au point.
4. Facilité "Debug": lors de la construction de l'application il est fréquent de rencontrer quelques difficultés au niveau du fonctionnement général. Cet outil nous permettra de suivre principalement l'étape d'initialisation de l'application afin d'aboutir à une coopération réaliste entre les objets. De même, cette fonction pourrait contribuer à la mise au point de l'application construite.

- Le niveau objet:

    1. Facilité "Link": cette fonction figure parmi les facilités de mise au point d'un objet qui vient d'être construit. En effet il est nécessaire d'attribuer à chaque objet un dispositif physique qu'il représentera. Cette opération serait d'une grande utilité pour mettre en évidence le deuxième mode de fonctionnement de MSS (par connection directe) mentionné au début de ce chapitre.
    2. Facilité "Sample": celle-ci pourrait être un outil d'aide, qui permet de fournir un modèle d'objets à suivre.

Conclusion

L'objectif initial de notre projet est de réaliser un outil graphique évolutif d'expertise et d'aide à la conduite, nommé MSS. Un tel outil nous permet d'exploiter la connaissance relative à certains problèmes de décision et de surveillance concernant les environnements basés sur des systèmes dynamiques répartis.

Nous avons présenté l'architecture de MSS en se basant sur ces deux sous-systèmes transformationnel et réactif. Dans le premier sous-système nous avons illustré la hiérarchie de MSS tandis que dans le second nous avons présenté les fonctionnalités de celui-ci afin de mieux illustrer la notion de décentralisation et d'interaction entre les différentes composantes du système simulé. Nous avons ensuite exposé les principes de la construction de l'interface graphique associé a MSS. Dans le dernier chapitre, nous avons présenté les démarches d'utilisation de MSS, en utilisant de deux exemples utilisés comme support d'argumentation.

Suite à notre étude et à la réalisation de MSS, il nous paraît important de mentionner les deux points principaux qui devraient être pris en compte dans les méthodes de

conception de tels systèmes.

1. Distinguer, au sein du système étudié, la partie transformationnelle de la partie réactive. Cette analyse permet de séparer deux modes de fonctionnement. La partie transformationnelle exige des mécanismes de coopération importants (transfert de messages riches en informations) et n'est généralement pas soumise à des contraintes temporelles critiques. La partie réactive, au contraire, est soumise à des contraintes temporelles critiques.
2. Décomposer la partie essentiellement transformationnelle en éléments fonctionnels. Ceci permet sa structuration en tâches et donne à l'outil un caractère modulaire.

Nous avons aussi eu l'occasion d'exploiter la boîte à outils GARNET. Celle-ci nous a été d'une grande utilité. Cependant, il faut signaler que nous aurions dû utiliser directement les outils graphiques CLIM, sans avoir recours à une autre surcouche. Malheureusement, ce dernier n'était pas à notre disposition au début de la réalisation.

Plusieurs recherches dans le domaine de l'algorithmique distribuée sont en cours au sein du laboratoire de Recherche en Informatique Fondamentale et Appliquée (RIFA) à l'Université de Sherbrooke. Celles-ci peuvent être un moyen d'enrichissement de notre système, en particulier de sa bibliothèque d'algorithmes de diagnostic, de surveillance et de prédiction.

Bibliographie

[1] P. K. Andow and F. P. Lees. Process computer alarm analysis: Outline of a method based on list processing. *Transactions of the Institute of Chemical Engineering 53*, pages 195–208, September 1975.

[2] C. André and L. Fancelli. étude d'une réalisation mixte d'un système temps réel. *Automatique Productique Informatique Industrielle*, 19(6):319–330, 1991.

[3] B. el. Ayeb. *Méthodes, langages et outils de spécification et construction des systèmes de diagnostic technique.* PhD thesis, CRIN / INRIA-Lorraine France, 1989.

[4] J. Coutaz. *Interface Homme-Ordinateur.* Dunod informatique, 1990.

[5] F. R. Hamilton et M. J. Kocurek. *Les fibres secondaires.* Cégep de trois-rivières, 1991.

[6] J. Ferber. *Les systèmes multi-agents.* Inter-Editions, 1995.

[7] A. Gram. *Raisonner pour programmer.* Dunod informatique, 1986.

[8] M. Iri, K. Aoki, E. O'Shima, and H. Matsuyama. An Algorithm for diagnosis of systems failures in the chemical process. *Computer & chemical engineering*, pages 489–493, September 1979.

[9] M. Iri, J. Shiozaki, E. O'Shima, and H. Matsuyama. An Improved Algorithm for diagnosis of systems failures in the chemical process. *Computer & chemical engineering*, pages 285–293, May 1979.

[10] J. H. Kleinau. Toward a pollution-free system of secondary fibre usage. *Canadian Pulp & Paper Association*, 3(2):26–29, 1971.

[11] C. Kolski. *Ingénierie des interfaces homme-machine.* Hermes, 1993.

[12] B. A. Myers, D. Giuse, A. Mickish, and B. V. Zanden. *The Garnet Reference Manuals.* Carnegie Mellon, 1993.

[13] A. Shraoui. Vers une approche globale de surveillance des systèmes à événements discrets. *Automatique Productique Informatique Industrielle*, 26(2):49–63, 1992.

[14] H. Thimbleby. *User Interface Design.* ACM press, 1990.

[15] V. Venkatasubramanian and S. H. Rich. An Object Oriented two-tier architecture for integrating completed and deep-level knowledge for process diagnosis. *Computer & chemical engineering*, pages 903–921, May 1988.